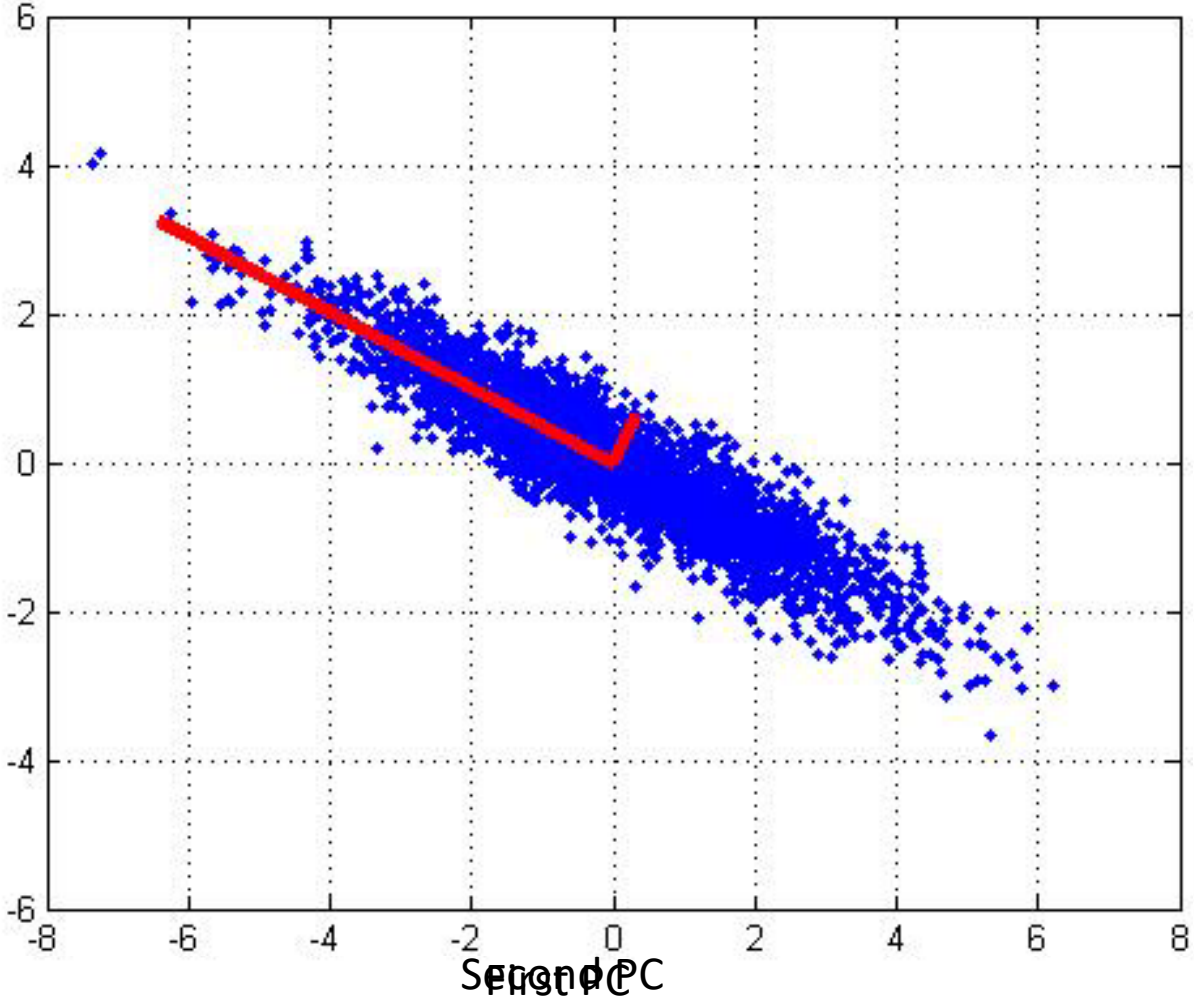
Autoencoders

Slide Credit: Piotr Mirowski, Oliver Durr, Diederik Kingma, Pascal Vincent and others

Unsupervised Representation Learning

- Principal Component Analysis (PCA)
- Sparse Representation
- Latent Factor Analysis

Principal Component Analysis (PCA)



- Prefer axes (orth. bases) in descending order of *variance* (information)

PCA algorithm I
(sample covariance matrix)

- Given data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute covariance matrix Σ

$$\boxed{\Sigma = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T}$$

where

$$\boxed{\bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i}$$

- **PCA** basis vectors = the eigenvectors of Σ

- Larger eigenvalue $\Rightarrow$ more important eigenvectors

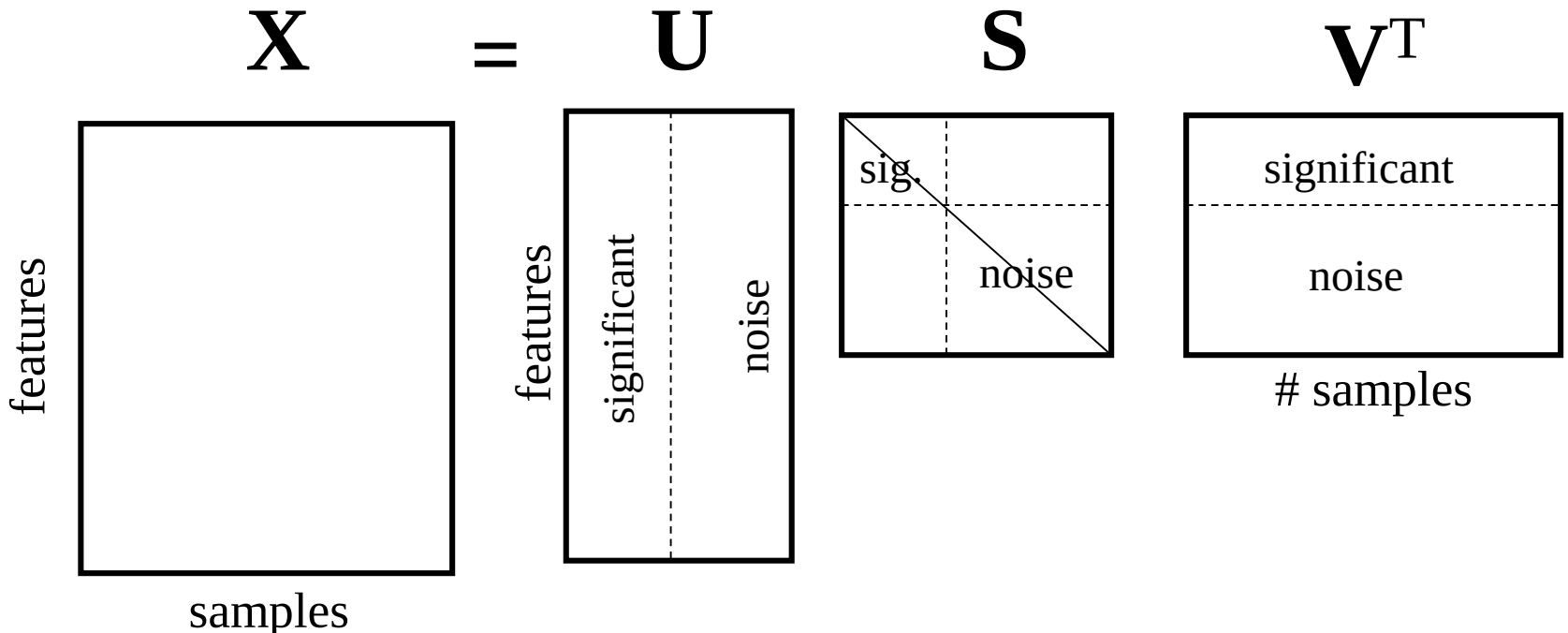
PCA algorithm II
(SVD of the data matrix)

Singular Value Decomposition of the **centered** data matrix **X**.

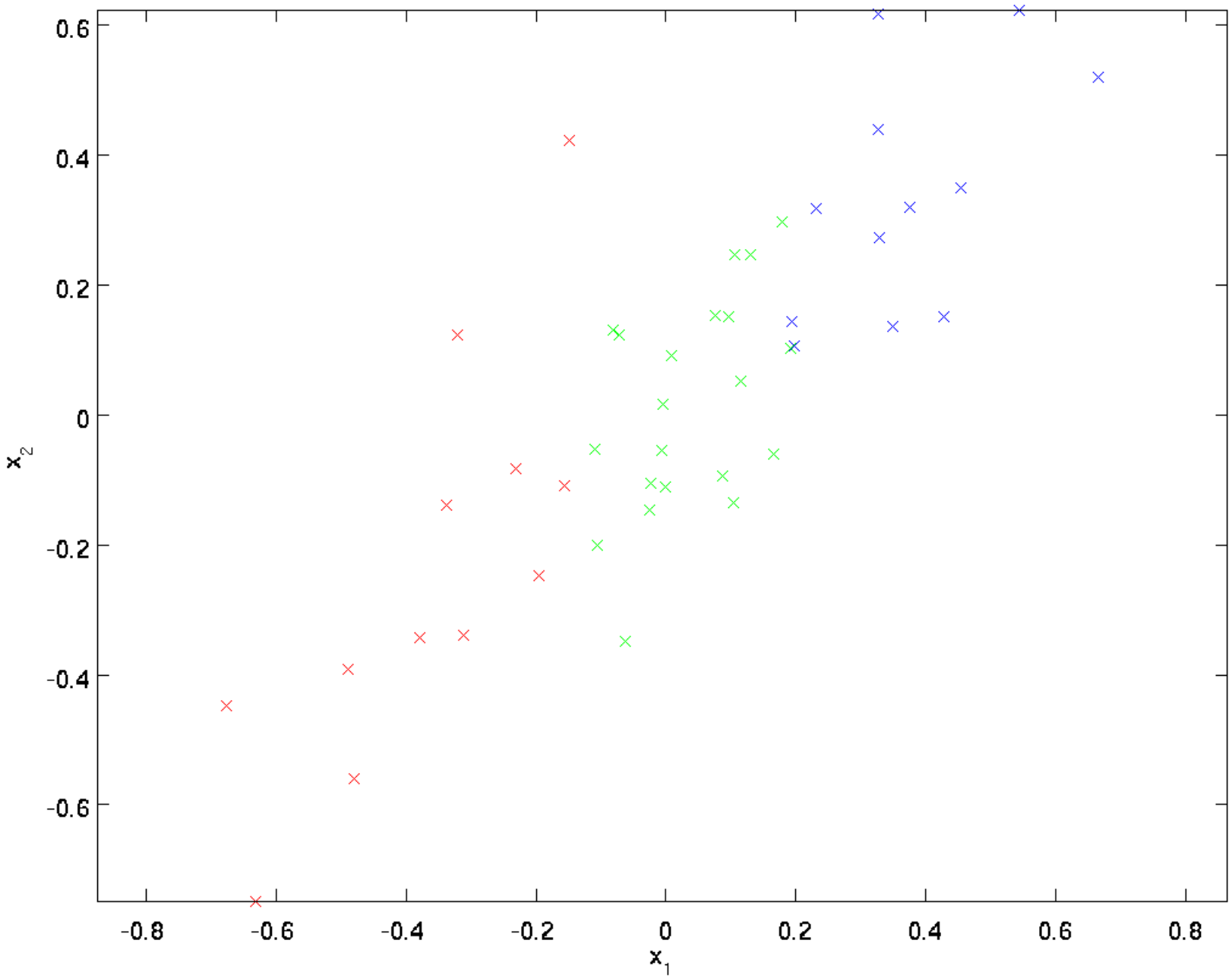
$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m] \in \mathbb{R}^{N \times m},$$

m: number of instances,
N: dimension

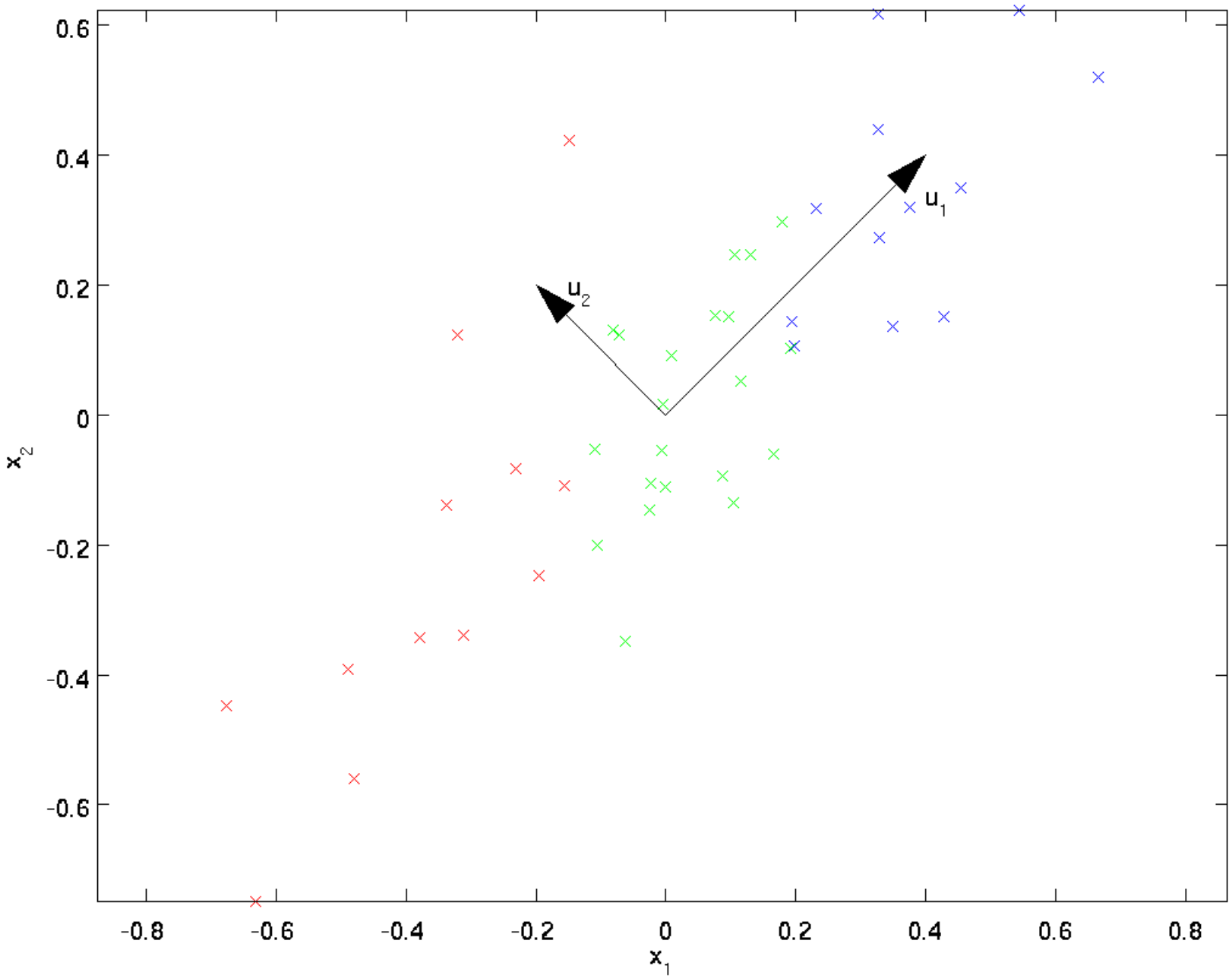
$$\mathbf{X}_{\text{features} \times \text{samples}} = \mathbf{USV}^{\mathrm{T}}$$



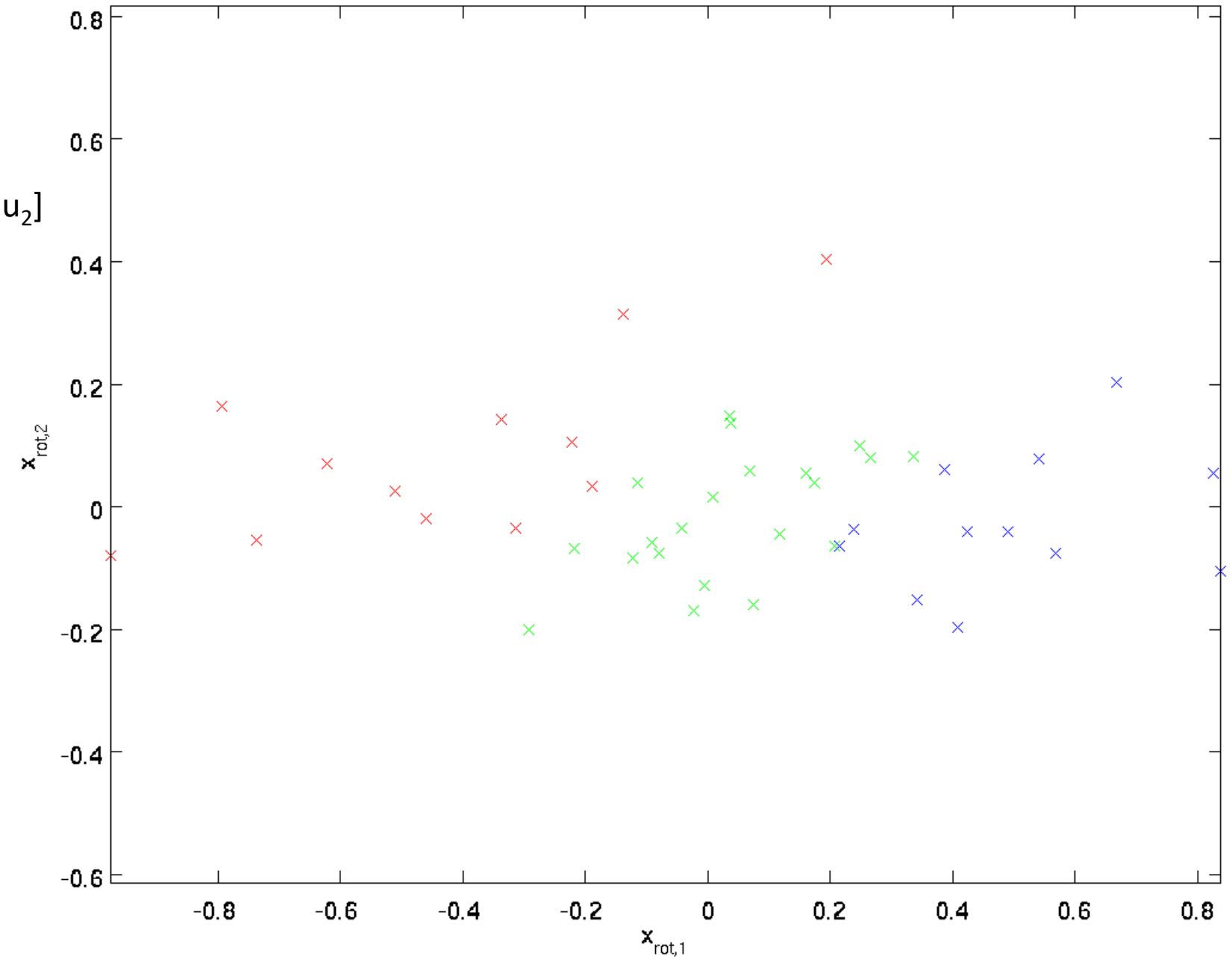
Raw Data



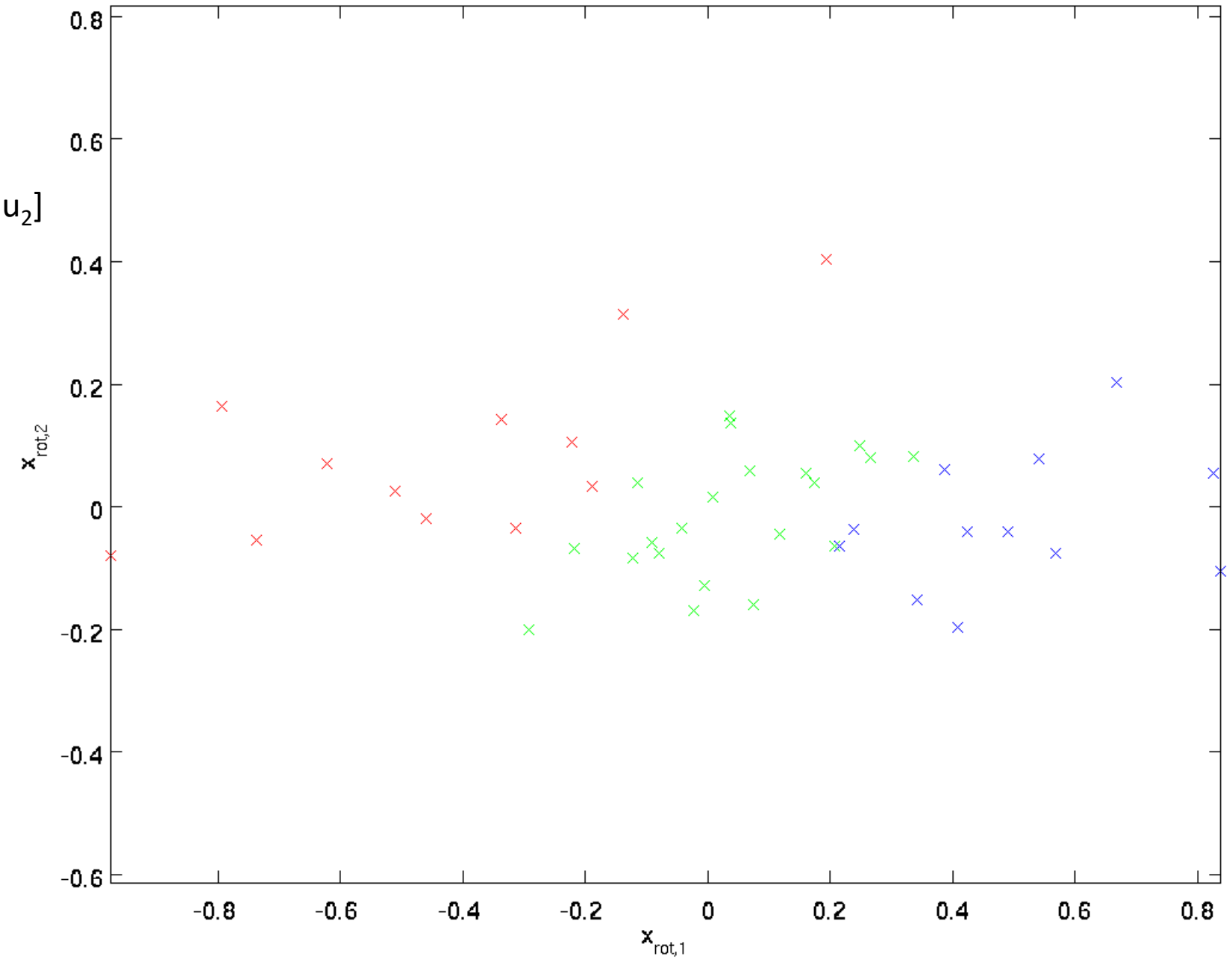
Apply PCA
Identify Principal Components



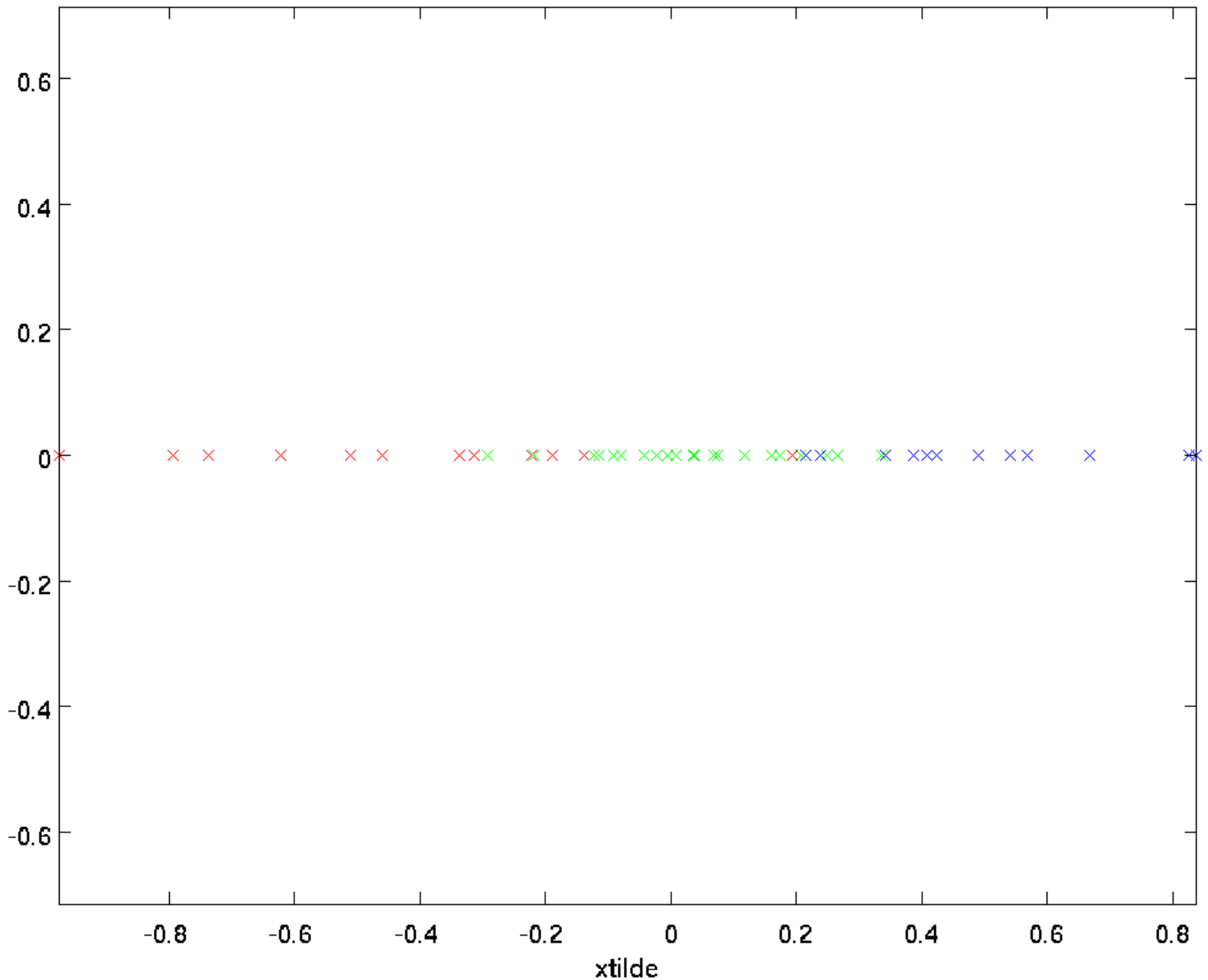
Rotate by $U = [u_1\ u_2]$



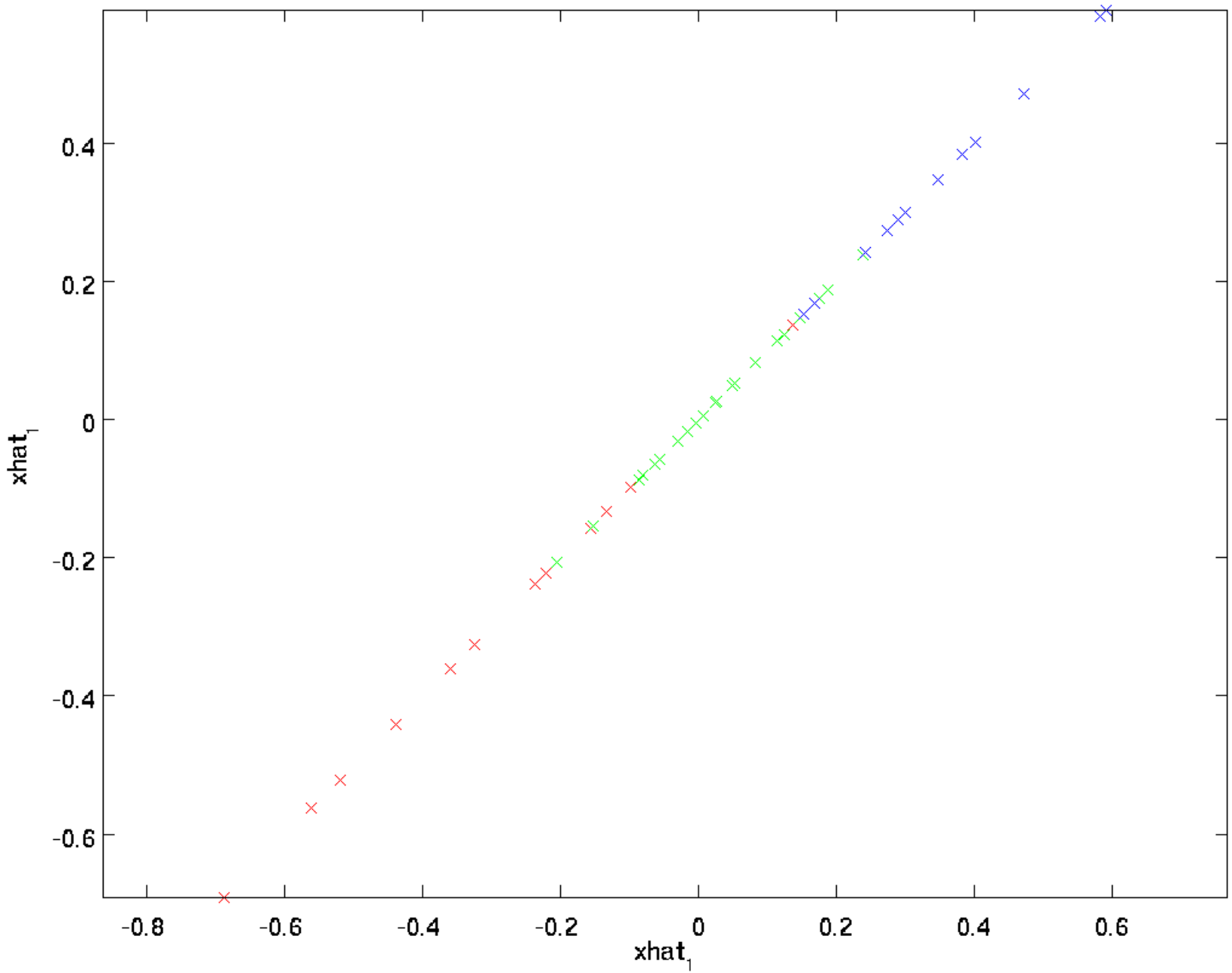
Rotate by $U = [u_1\ u_2]$



Project on u_1
Reduced Dimensionality



Reconstruct



PCA

- Low Dimensional Representation (Dimensionality Reduction)

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]$$

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$$

$$\mathbf{W} = \mathbf{U}_{d \times k}, \quad k \leq d$$

$$\mathbf{b} = \frac{1}{m} \sum_i \mathbf{x}_i$$

$$\mathbf{h}_i = \mathbf{U}^\top \mathbf{x}_i = \mathbf{S}\mathbf{v}_i$$

Can use for dim. reduction

- Reconstruction

$$\mathbf{x} = \mathbf{W}\mathbf{h} + \mathbf{b}$$

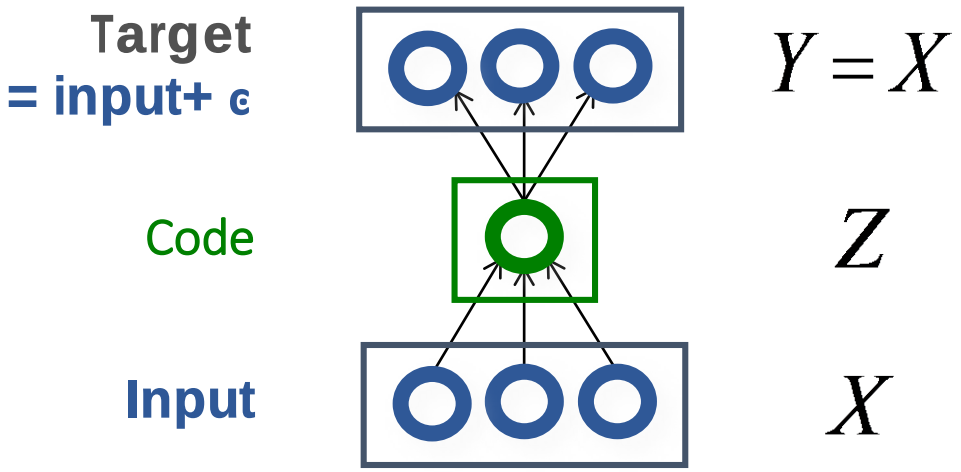
PCA as a Generative Model

- If we know **h**, **W** and **b** we can generate **x**

$$\mathbf{x} = f(\mathbf{h}) = \mathbf{W}\mathbf{h} + \mathbf{b}$$

- Can a neural network be designed to implement PCA?
  - PCA minimizes the L_2 reconstruction error => loss function for neural net.

PCA: Auto-encoder with linear activation



"Bottleneck" code
i.e., low-dimensional,
typically dense,
distributed
representation

Sparse Coding

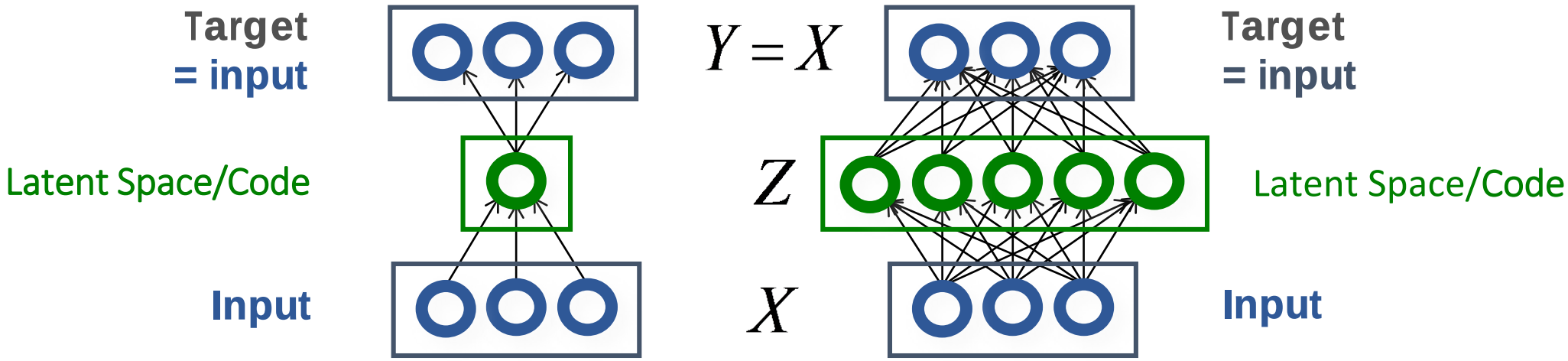
$$p(\boldsymbol{x} \mid \boldsymbol{h}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{W}\boldsymbol{h} + \boldsymbol{b}, \frac{1}{\beta}\boldsymbol{I}).$$

$$p(h_i) = \text{Laplace}(h_i; 0, \frac{2}{\lambda}) = \frac{\lambda}{4} e^{-\frac{1}{2}\lambda|h_i|}$$

$$\arg\min_{\boldsymbol{h}} \lambda||\boldsymbol{h}||_1 + \beta||\boldsymbol{x} - \boldsymbol{W}\boldsymbol{h}||_2^2,$$

(Goodfellow 2016)

Auto-encoder



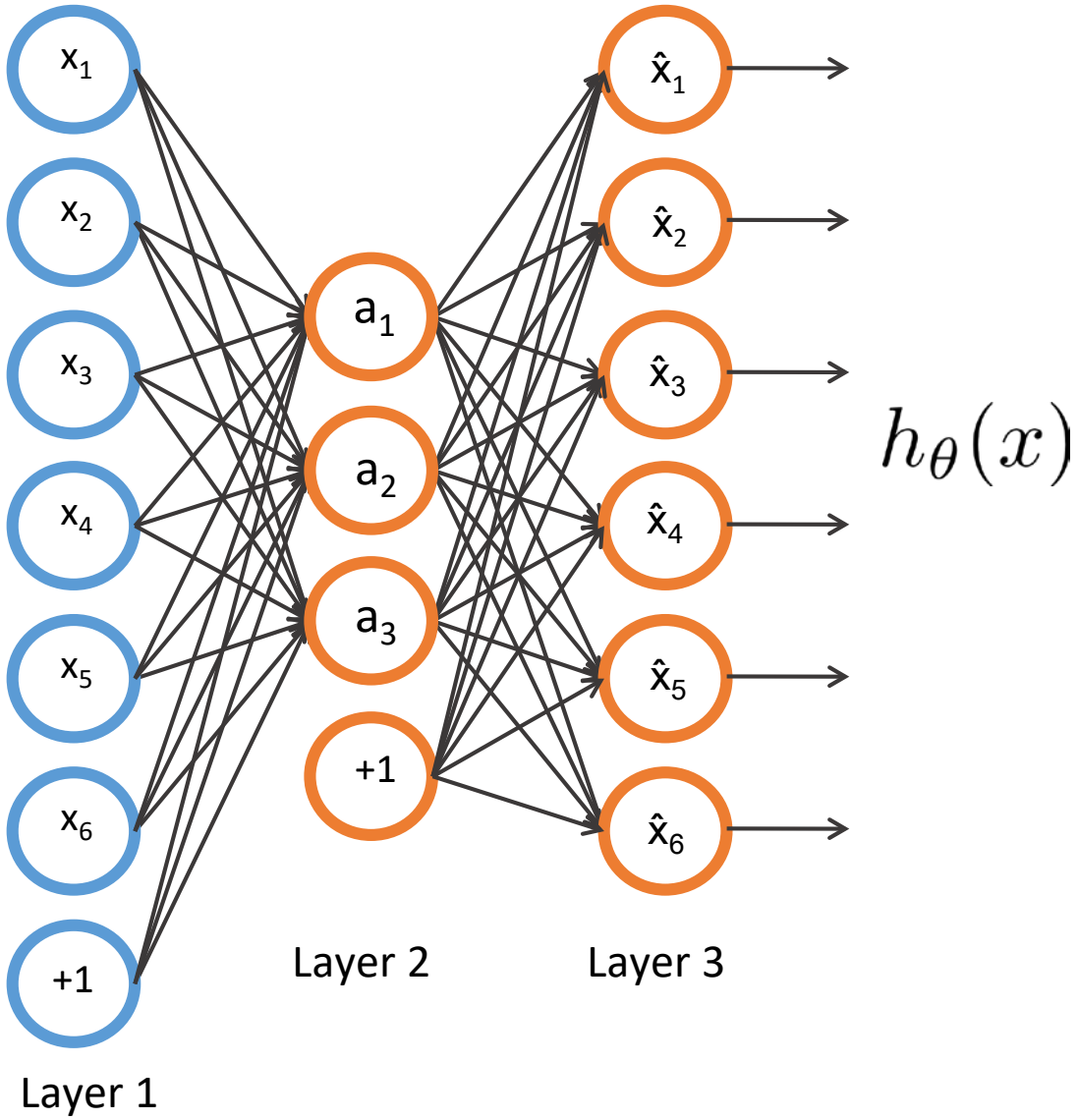
"Undercomplete" code
i.e., low-dimensional,
typically dense,
distributed
representation

"Overcomplete" code
i.e., high-dimensional,
always sparse,
distributed
representation

Applications

- Pretraining Supervised Classifiers
  - Initialization of Deep Architectures
- Feature Representation Learning
  - Stacked AE
  - Denoising AE
- Generative Modeling (Learning Data Distributions)
  - Variational AE

Stacked AE



Autoencoder.

Network is trained to output the input (learn identify function).

$$h_\theta(x) \simeq x$$

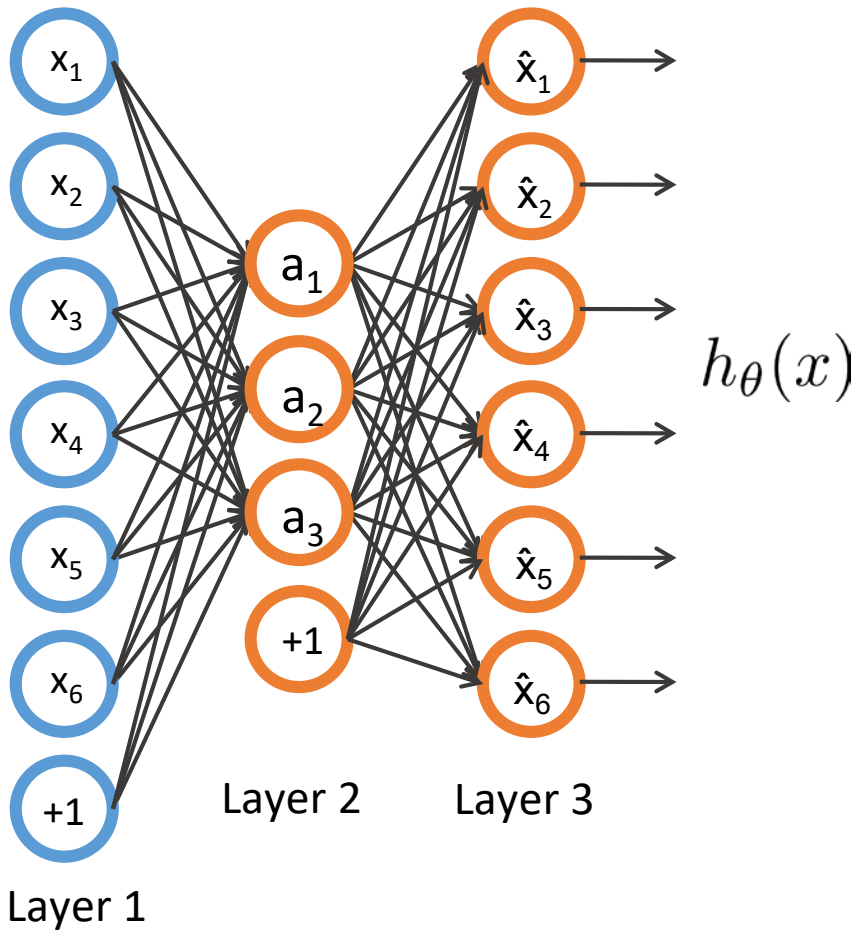
Trivial solution unless:

- Constrain number of units in Layer 2 (learn compressed representation), or
- Constrain Layer 2 to be sparse.

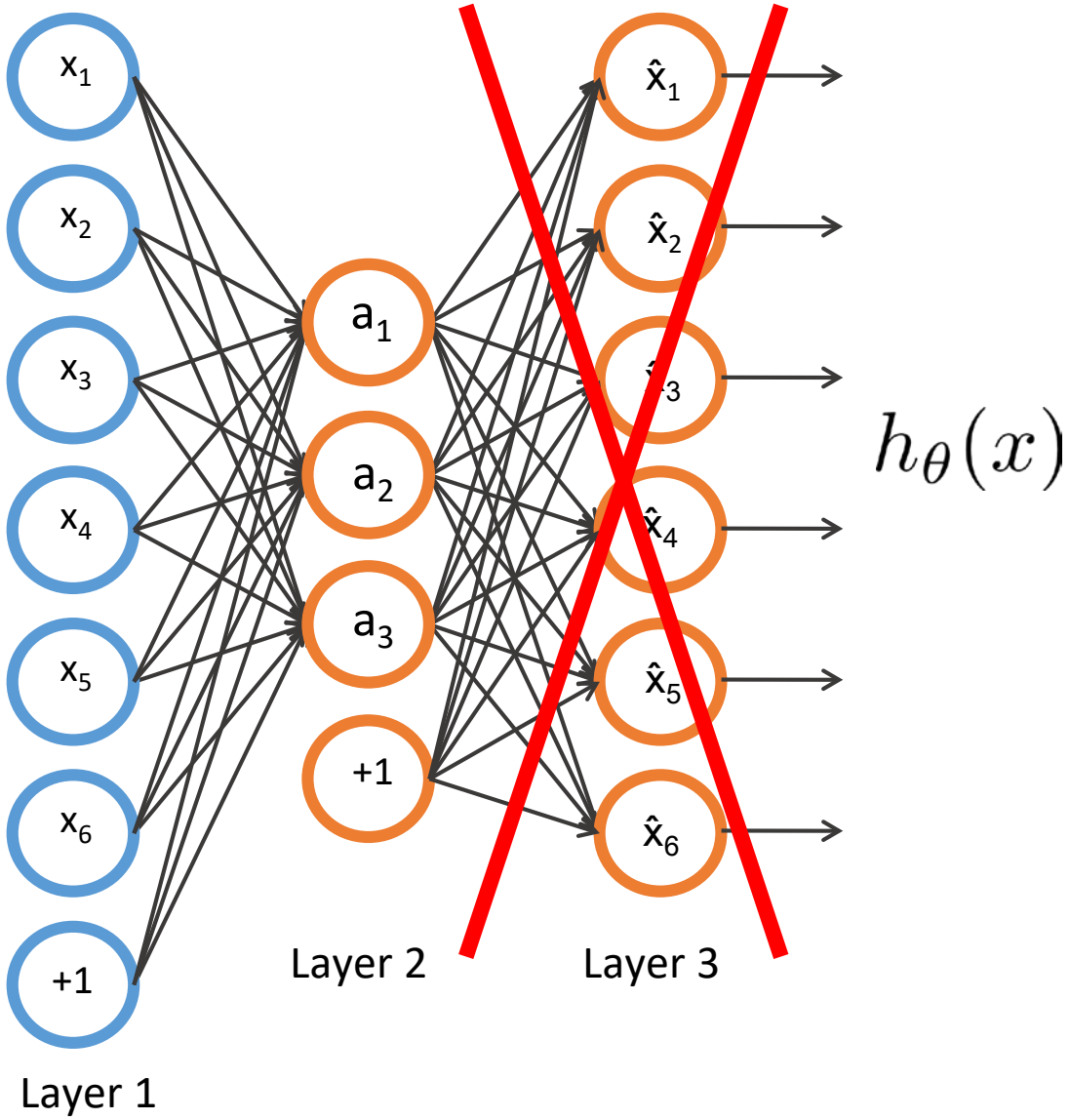
Stacked AE

- Loss Function for a sparse AE

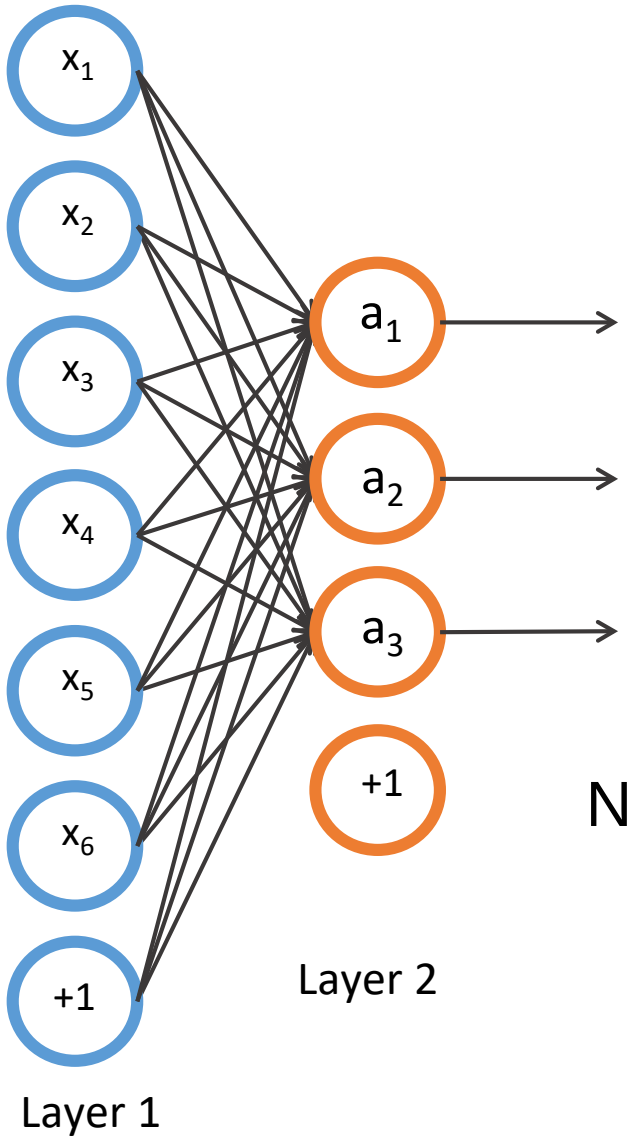
$$\min_\theta \underbrace{||h_\theta(x) - x||_2^2}_{\text{Reconstruction error term}} + \underbrace{\lambda||a||_1}_{L_1 \text{ sparsity term}}$$



Stacked AE

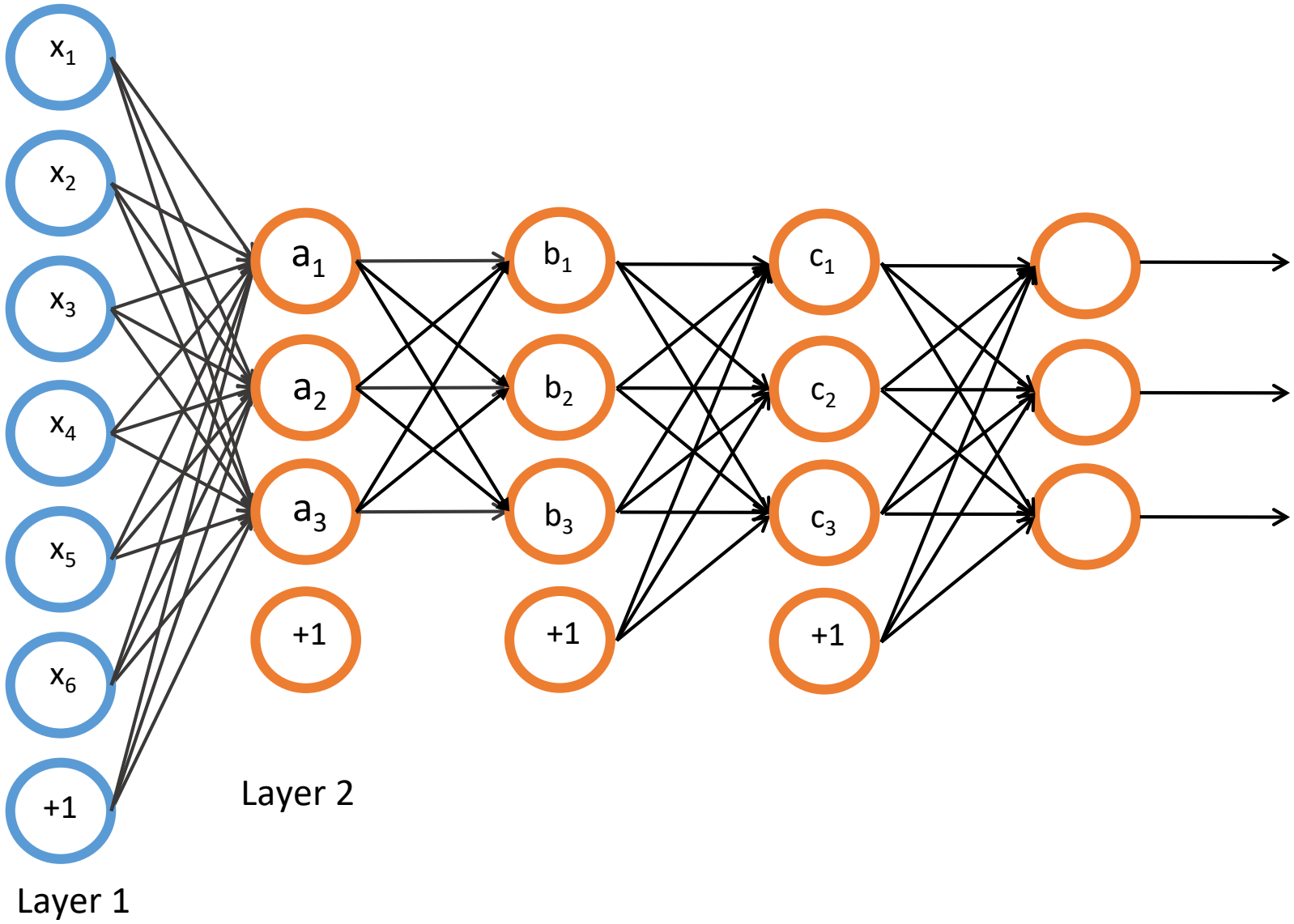


$$h_\theta(x)$$



$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

New representation for input.


x_1
x_2
x_3
x_4
x_5
x_6
+1
Layer 1
a_1
a_2
a_3
+1
Layer 2
b_1
b_2
b_3
+1
c_1
c_2
c_3
+1

Autoencoders

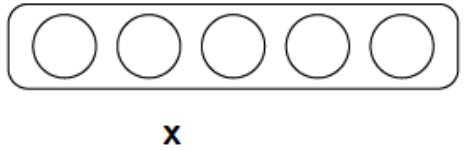
- Useful property
  - Intermediate representation
  - Deep AE → hierarchical (more abstract) representation

- Trivial solution is avoided by regularization

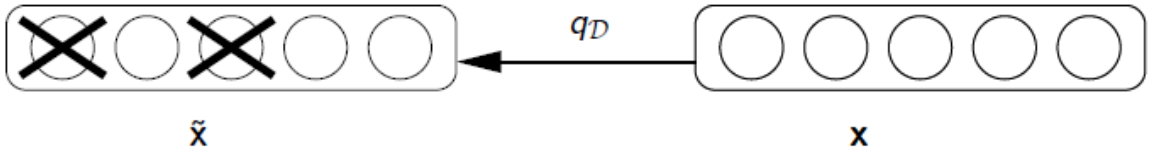
- Key idea is to retain information
  - Robustness to corruption of input?

Denoising AE

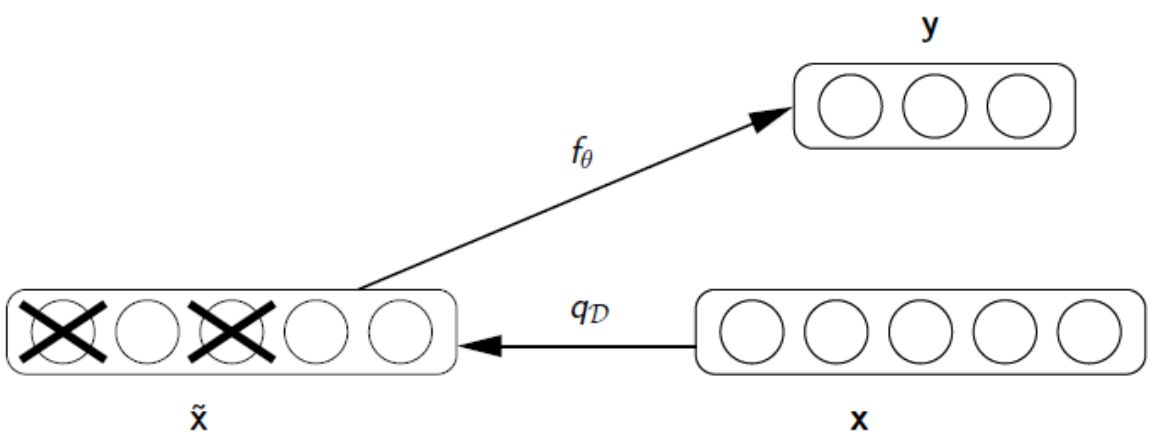
- Corrupt part of the input (make it zero, add noise, just about any kind of corruption)
- Train the AE to yield the uncorrupted input
- Stacking Denoising AE
  - Layer-wise training
  - For each hidden layer, uncorrupted input is used
  - Different from dropout… How?
  - Different from data augmentation… How?
- Form of variational inference in a generative model
  - Variational => approximate inference using a simpler approximate posterior of the latent variables
- Bridge Gap between DBNs and SAE
  - DBNs benefit from generative modelling capability of RBMs



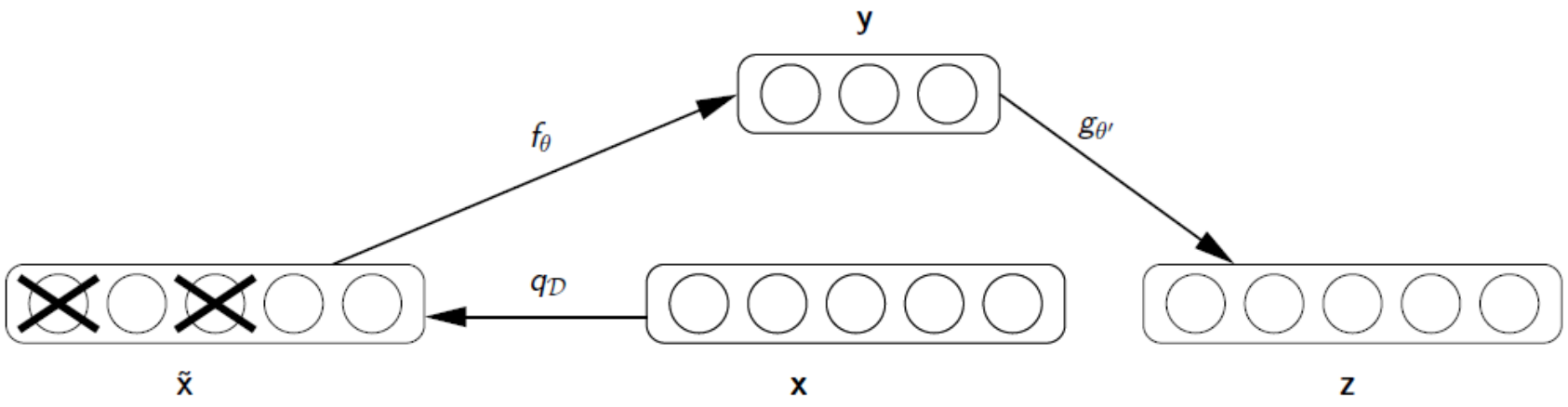
- Clean input $\mathbf{x} \in [0,1]^d$ is partially destroyed, yielding corrupted input: $\tilde{\mathbf{x}} \sim q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x})$.
- $\tilde{\mathbf{x}}$ is mapped to hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}})$.
- From $\mathbf{y}$ we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y})$.
- Train parameters to minimize the cross-entropy "reconstruction error"



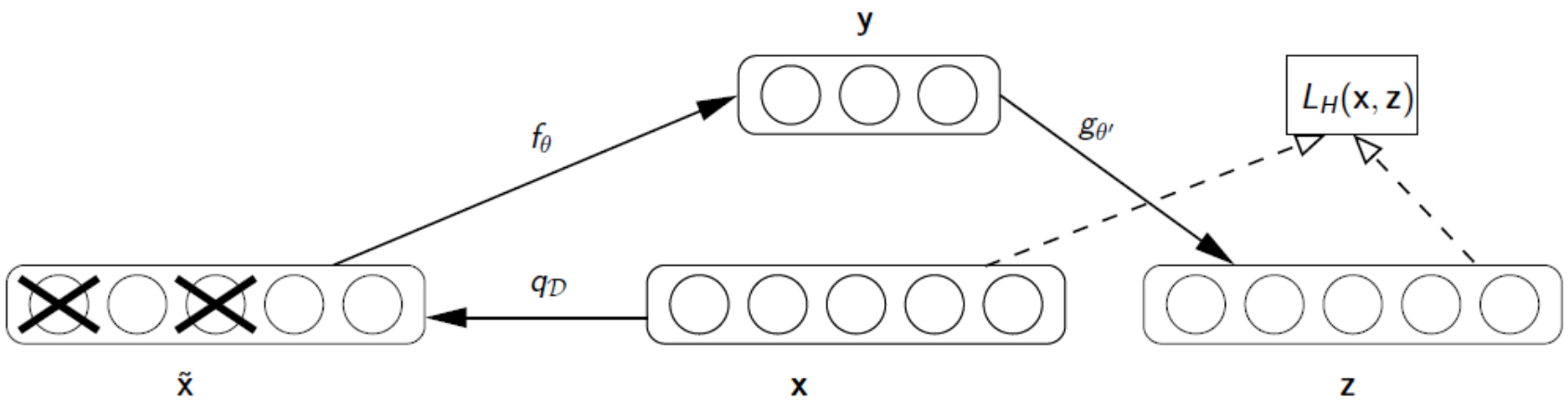
- Clean input $\mathbf{x} \in [0,1]^d$ is partially destroyed, yielding corrupted input: $\tilde{\mathbf{x}} \sim q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x})$.
- $\tilde{\mathbf{x}}$ is mapped to hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}})$.
- From $\mathbf{y}$ we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y})$.
- Train parameters to minimize the cross-entropy "reconstruction error"



- Clean input $\mathbf{x} \in [0,1]^d$ is partially destroyed, yielding corrupted input: $\tilde{\mathbf{x}} \sim q_\mathcal{D}(\tilde{\mathbf{x}}|\mathbf{x})$.
- $\tilde{\mathbf{x}}$ is mapped to hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}})$.
- From $\mathbf{y}$ we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y})$.
- Train parameters to minimize the cross-entropy "reconstruction error"



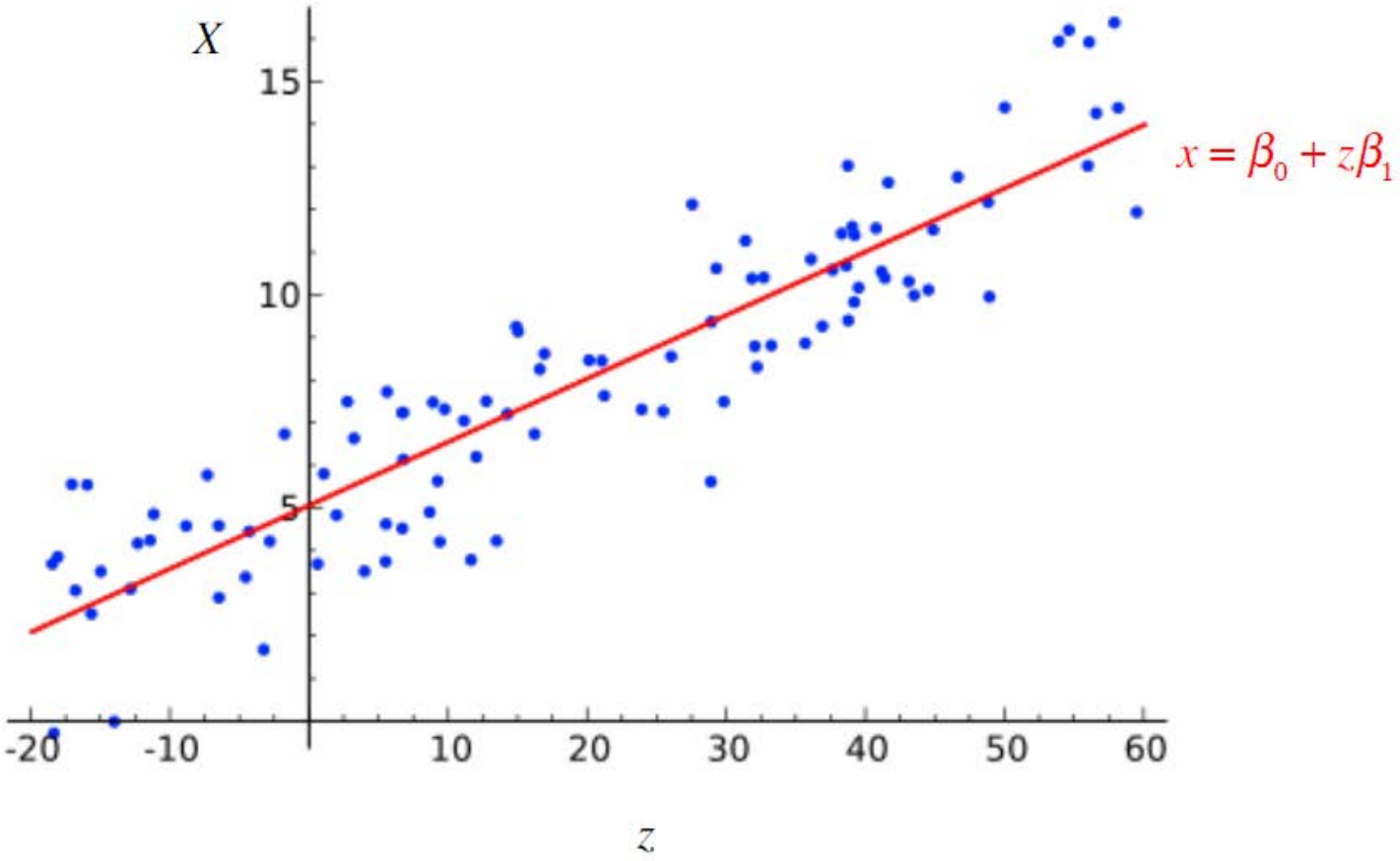
- Clean input $\mathbf{x} \in [0,1]^d$ is partially destroyed, yielding corrupted input: $\tilde{\mathbf{x}} \sim q_{\mathcal{D}}(\tilde{\mathbf{x}}|\mathbf{x})$.
- $\tilde{\mathbf{x}}$ is mapped to hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}})$.
- From $\mathbf{y}$ we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y})$.
- Train parameters to minimize the cross-entropy "reconstruction error"



- Clean input $\mathbf{x} \in [0,1]^d$ is partially destroyed, yielding corrupted input: $\tilde{\mathbf{x}} \sim q_\mathcal{D}(\tilde{\mathbf{x}}|\mathbf{x})$.
- $\tilde{\mathbf{x}}$ is mapped to hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}})$.
- From $\mathbf{y}$ we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y})$.
- Train parameters to minimize the cross-entropy "reconstruction error"

Generative Modeling using AE

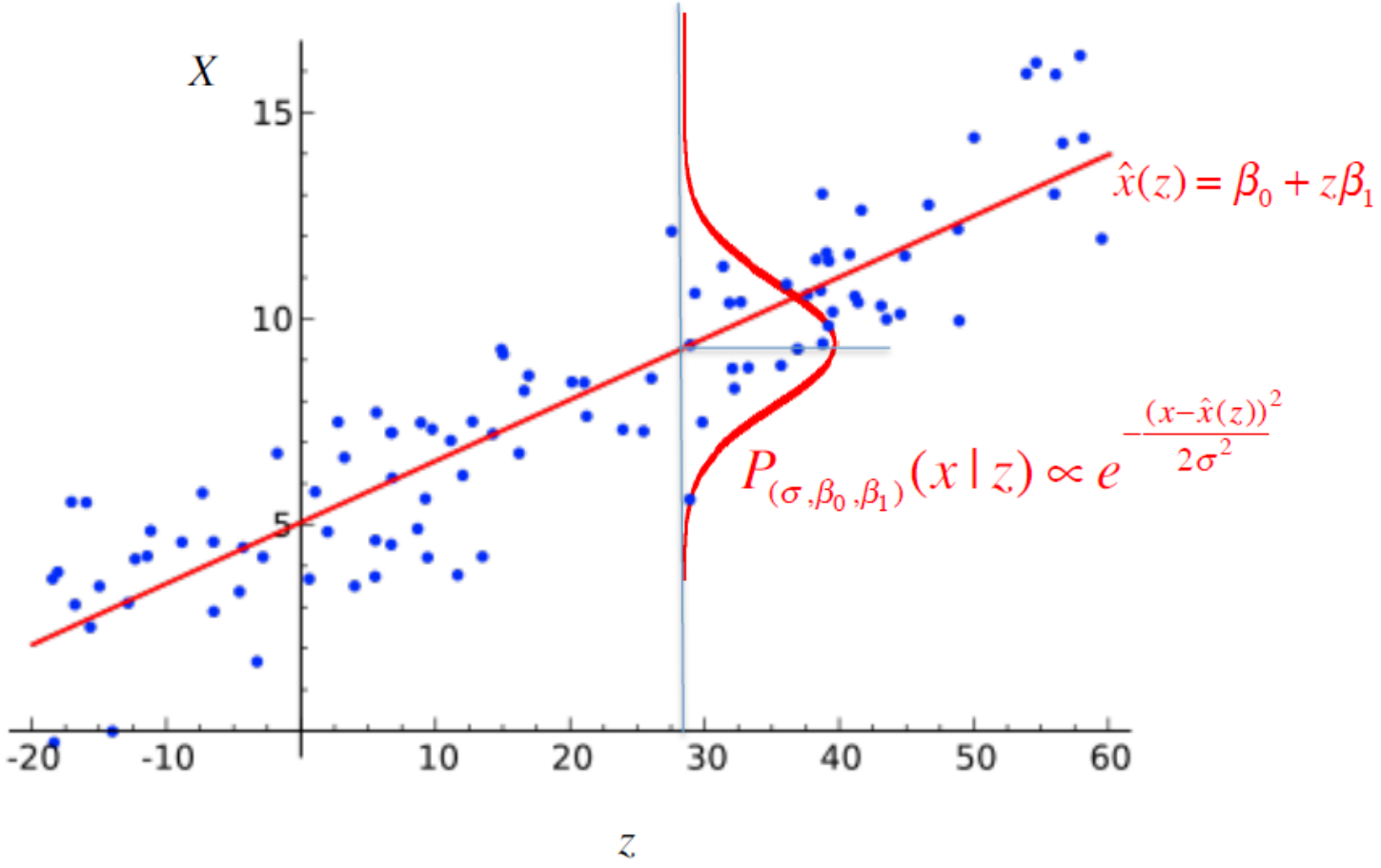
Linear Regression



PCA?

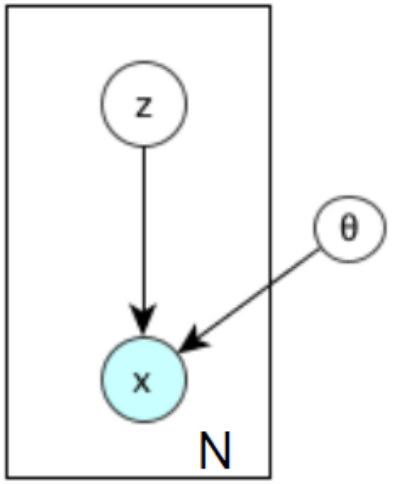
Error Modeling in Linear Regression

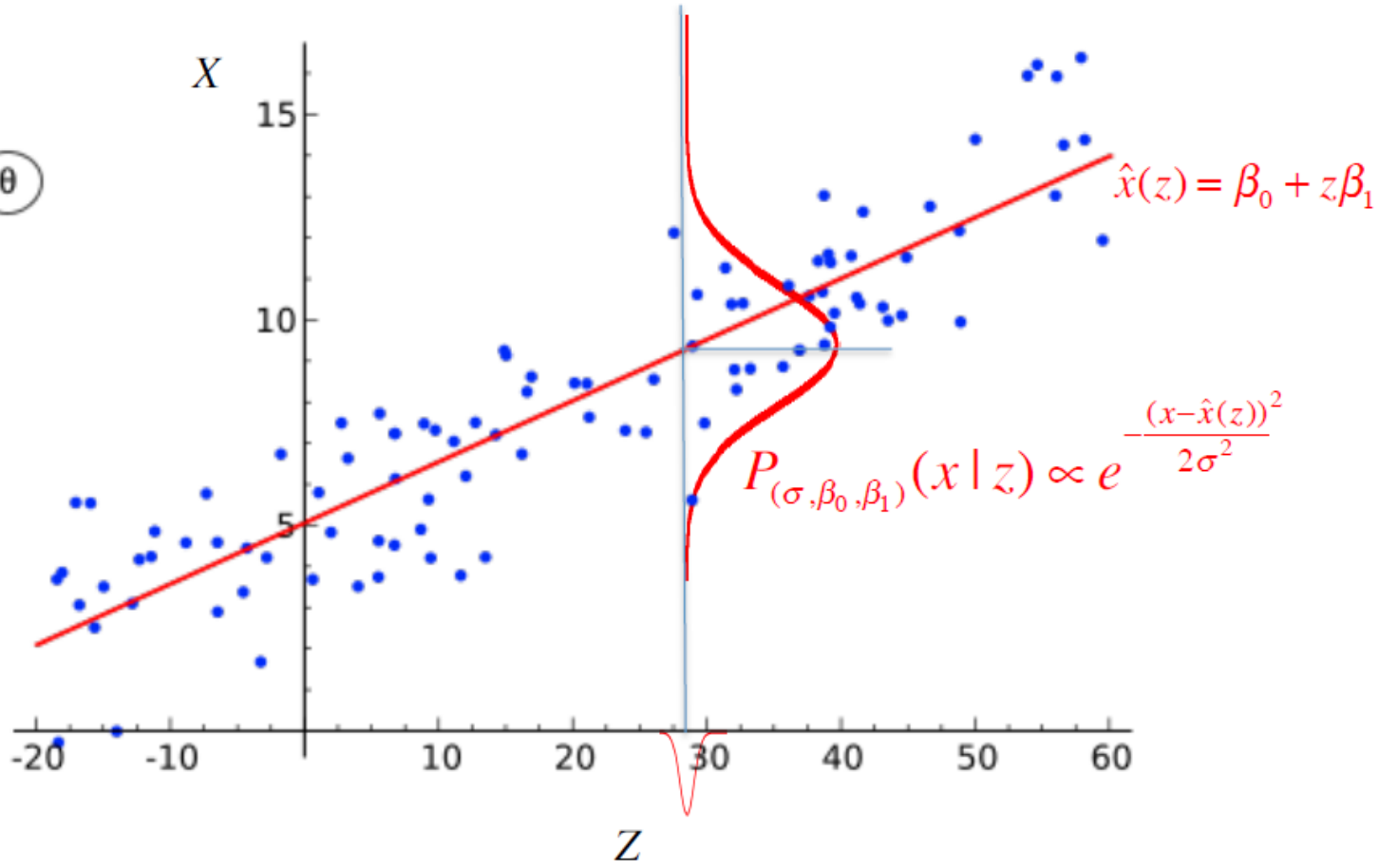
Statisticians additionally have $P_\theta(X \mid Z)$



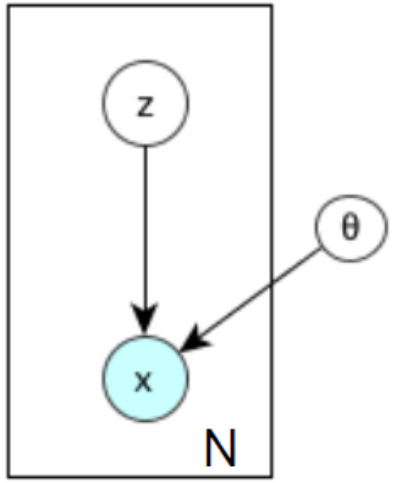
Linear Regression as a Graphical Model

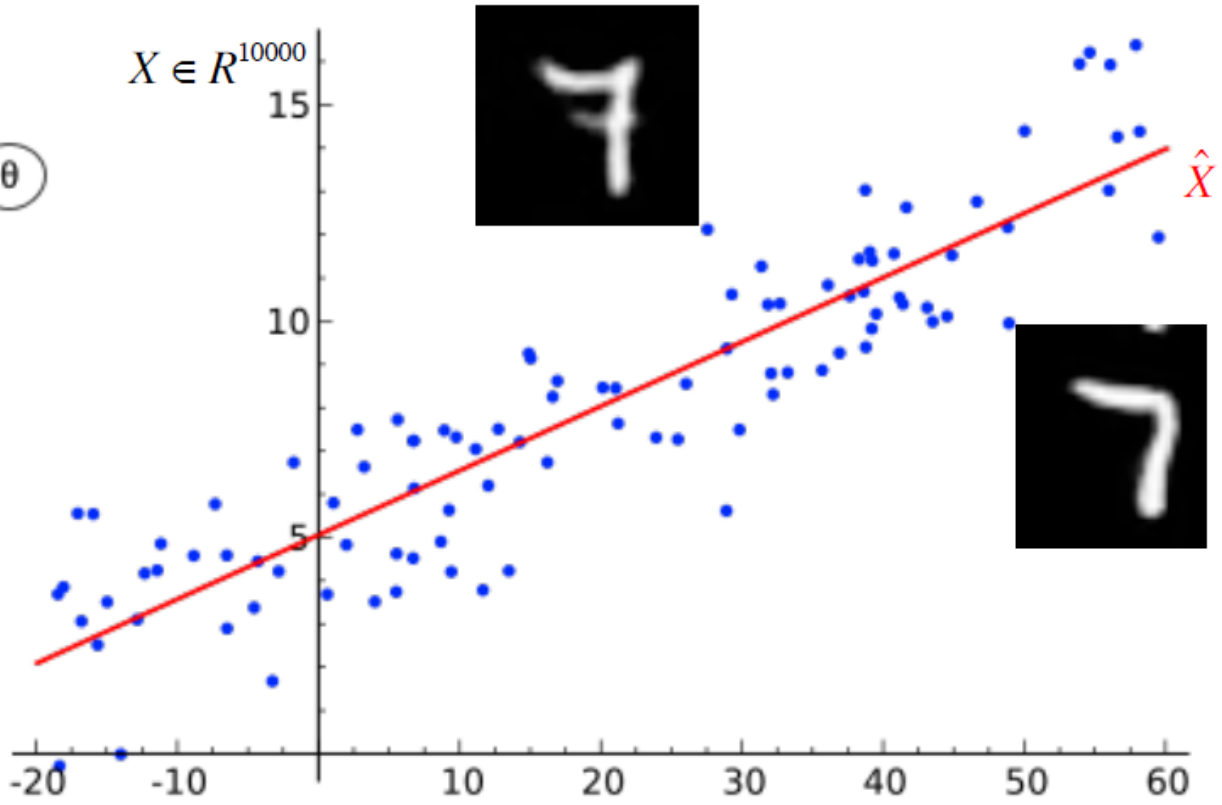
Statisticians additionally have $P_\theta(X \mid Z)$





Generating in Higher Dimensions





$Z \in R^n$ typically $n = 2,\ldots,50$ "latent space"

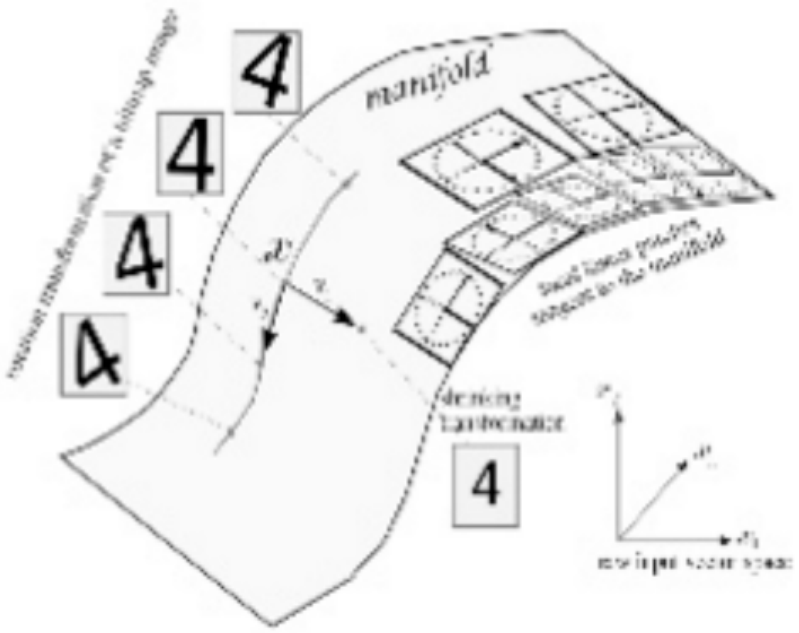
Is R^2 "big enough" to create images from R^{100000}?...

Generative Models for MNIST

http://www.dpkingma.com/sgvb_mnist_demo/demo.html

Manifold Learning

- X high dimensional vector
- Data is concentrated around a low dimensional manifold



- Hope finding a representation Z of that manifold.

1D

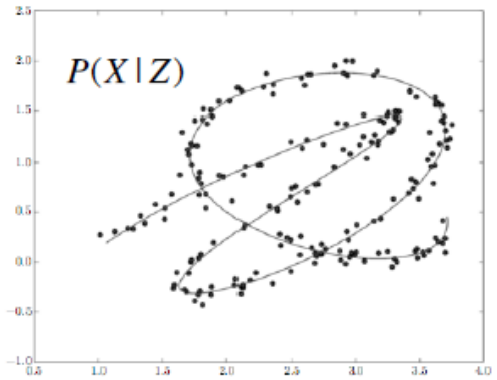
Low Dimensional representation a line

z_1

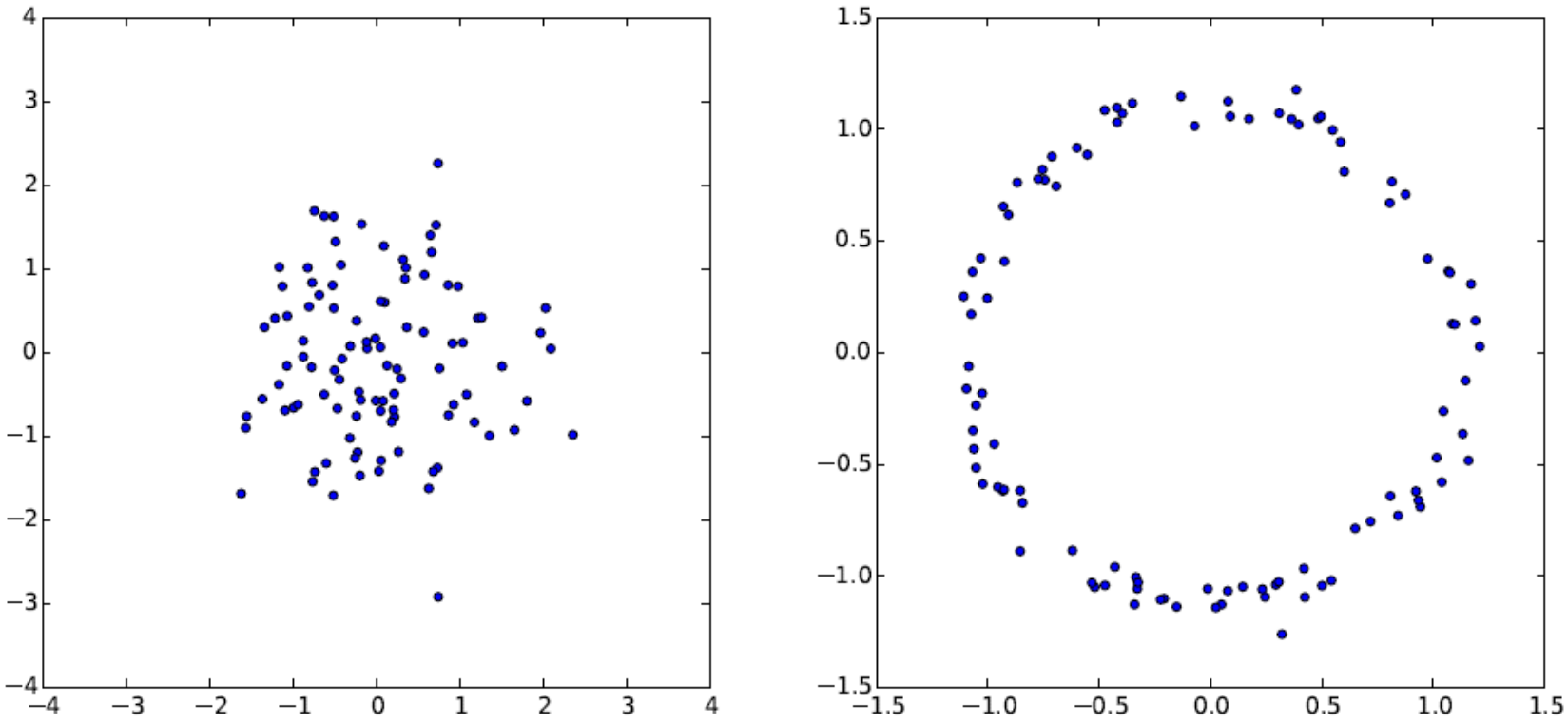
2D

High Dimensional (number of pixels)

x_2



x_1



$p(z_i) \sim \mathcal{N}(0, 1)$

$\boxed{X = g(z) = z/10 + z/||z||}$

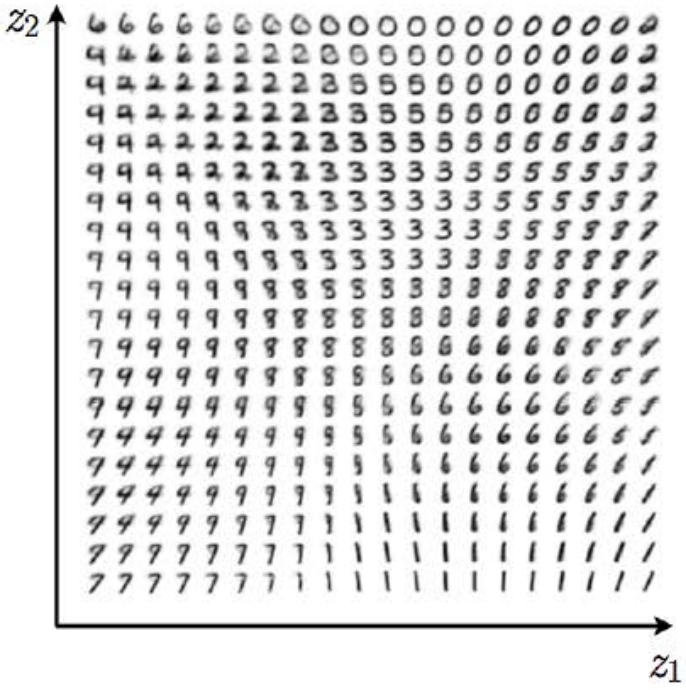
Can make this function arbitrarily complex

Figure 2: Given a random variable z with one distribution, we can create another random variable $X = g(z)$ with a completely different distribution. Left: samples from a gaussian distribution. Right: those same samples mapped through the function $g(z) = z/10 + z/||z||$ to form a ring. This is the strategy that VAEs use to create arbitrary distributions: the deterministic function g is learned from data.

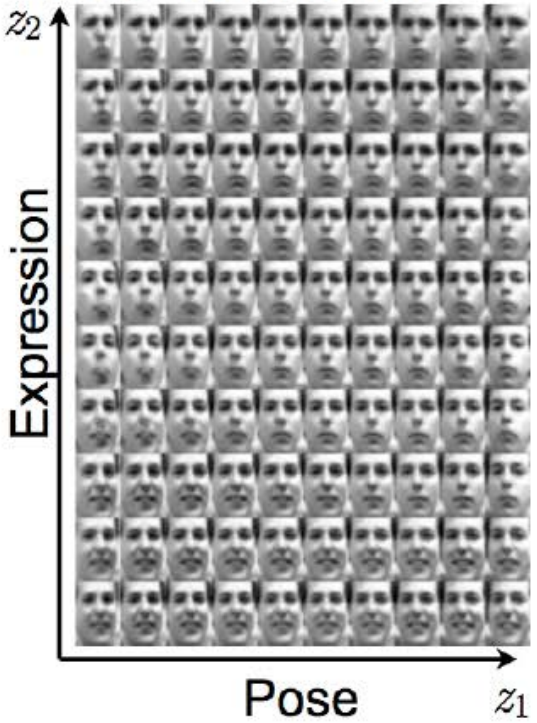
2-D Manifolds of Real Data

Examples of successful unfolding (2D $\rightarrow R^{28x28}$, R^{20x26})

MNIST:



Frey Face dataset:



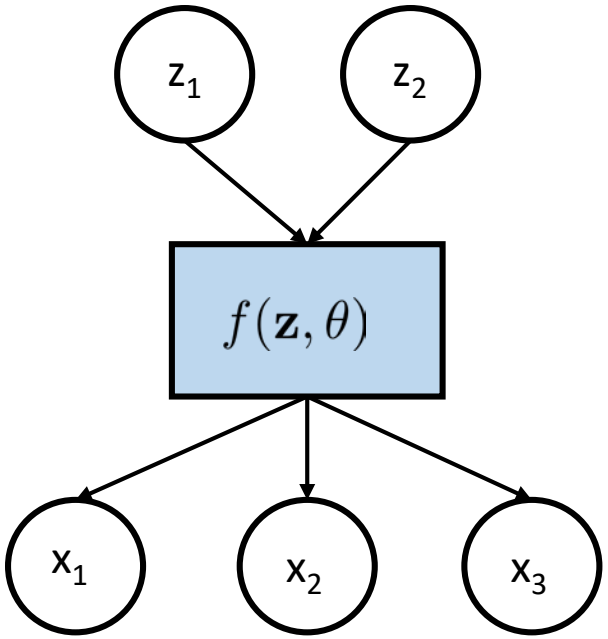
Frey Face dataset

2000 pictures of Brendan Frey (20x26)

Generated images: NOT part of the training set

Kingma and Welling. "*Auto-Encoding Variational Bayes, International Conference on Learning Representations." ICLR,* 2014

Generative Models



Latent Factors/Variables

$$\mathbf{z} \sim p(\mathbf{z}) = \prod_i p(z_i)$$

Parametric distribution for latent var.
Often assumed ind. → joint distr. is product

Observations

Assumed to have a more complicated distr.
Independent when conditioned on latent factors

$$\mathbf{x} = f(\mathbf{z}, \theta) + \text{noise}$$

$$e.g.\ \mathbf{x} = \mathbf{W}(\theta)\ \mathbf{z} + \mathbf{b}(\theta) + \text{noise}$$

Generative Models

- Generate z from the prior distribution $\mathbf{z} \sim p(\mathbf{z})$
- Generate x from the conditional distribution $\mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z})$

- Assumptions:
  - Both pdfs come from parametric families
  - pdfs are differentiable, both w.r.t. z and θ

- Unknowns:
  - True parameters θ
  - Values of the latent variables z

Generative Modeling

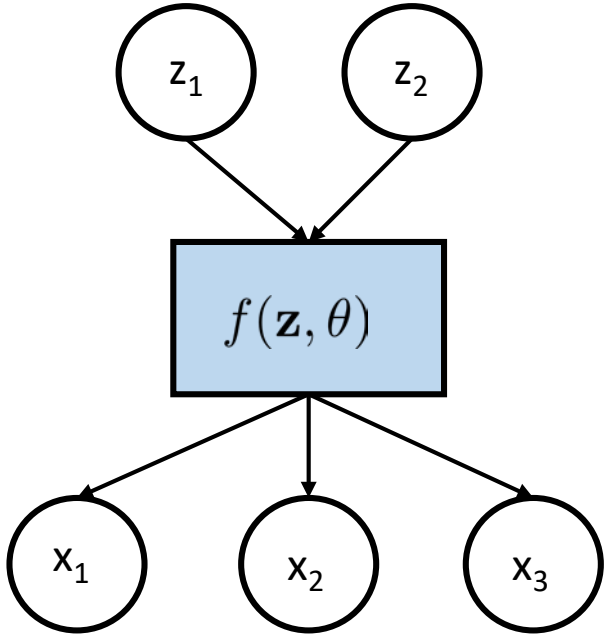
- Data has an unknown distribution $x \sim p_{gt}(X)$
- If $p(X)$ approximates $p_{gt}(X)$, we can generate $x' \sim p(X)$ similar to x

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

- Challenges
  - x could be very high dimensional (e.g., images)
  - p(x) is usually intractable (multi-dimensional integral)
  - p(z|x) is also intractable

- Variational AE try to embed this complexity in $f(\mathbf{z},\boldsymbol{\theta})$

$$p(z_i) \sim \mathcal{N}(0,1)$$



$$\mathbf{x} = f(\mathbf{z},\theta) + \text{noise}$$

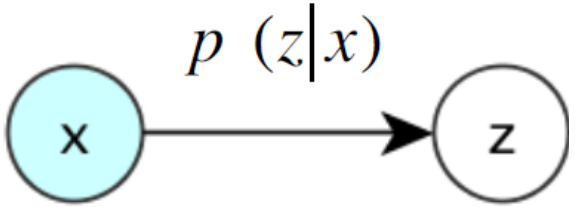
$$\sim \underbrace{\mathcal{N}(X|f(\mathbf{z};\theta),\sigma^2 I)}$$

Gaussian with mean $f(\mathbf{z},\theta)$ and cov. $\sigma^2 I$

Generative Modeling

- Goal:
  - **efficiently learn** approx. <u>maximum likelihood</u> parameters θ
    - Enables us to model the underlying random process
    - Generate new data samples
  - **efficiently infer** latent variables for new observations x
    - Data representations and/or coding
  - **efficiently approximate** the marginal inference of the variable x
    - perform inference related tasks on x, e.g., inpainting, superresolution,

Variational Inference



Training use maximum likelihood
of p(x) given the training data

Problem:

$$p\ (z|x)$$

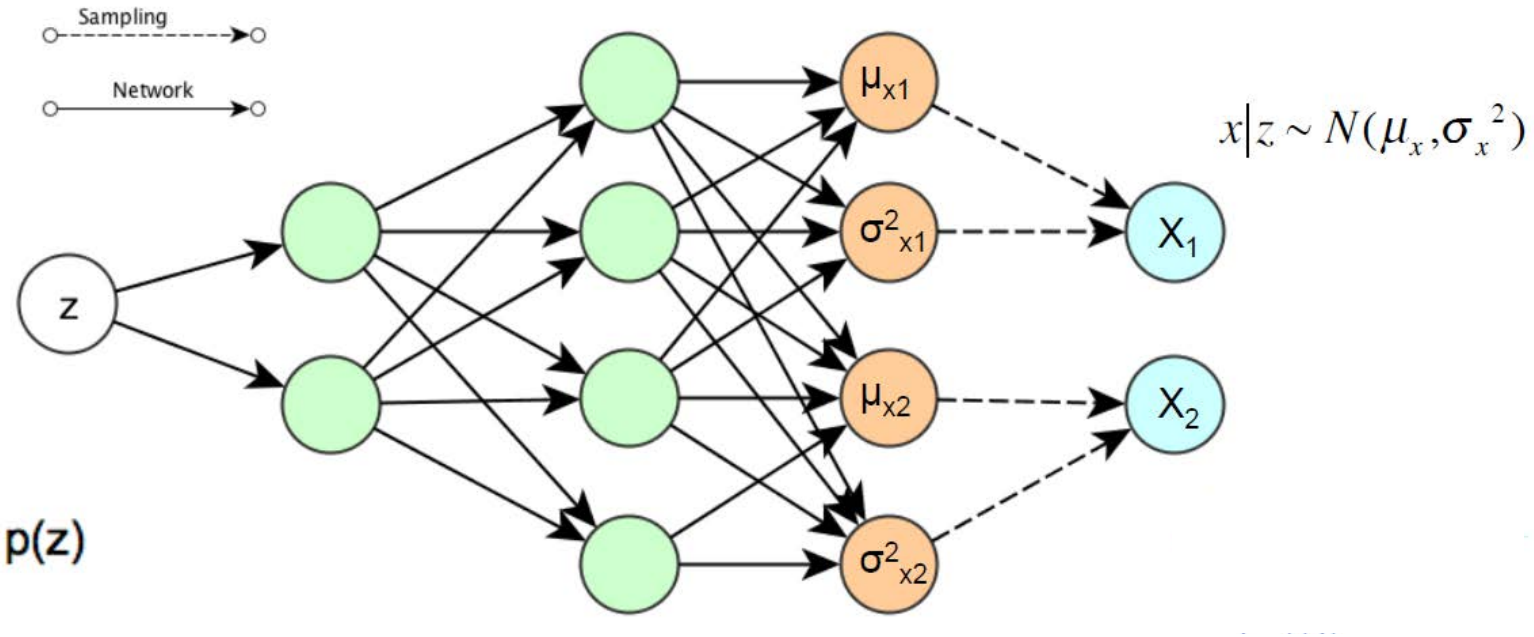
Cannot be calculated:

Solution:
Markov Chain Monte Carlo (MCMC): asymptotically unbiased,
but takes too many iterations of sampling

Solution:
approximate by $q_\phi(z|x)$

Model for Decoder

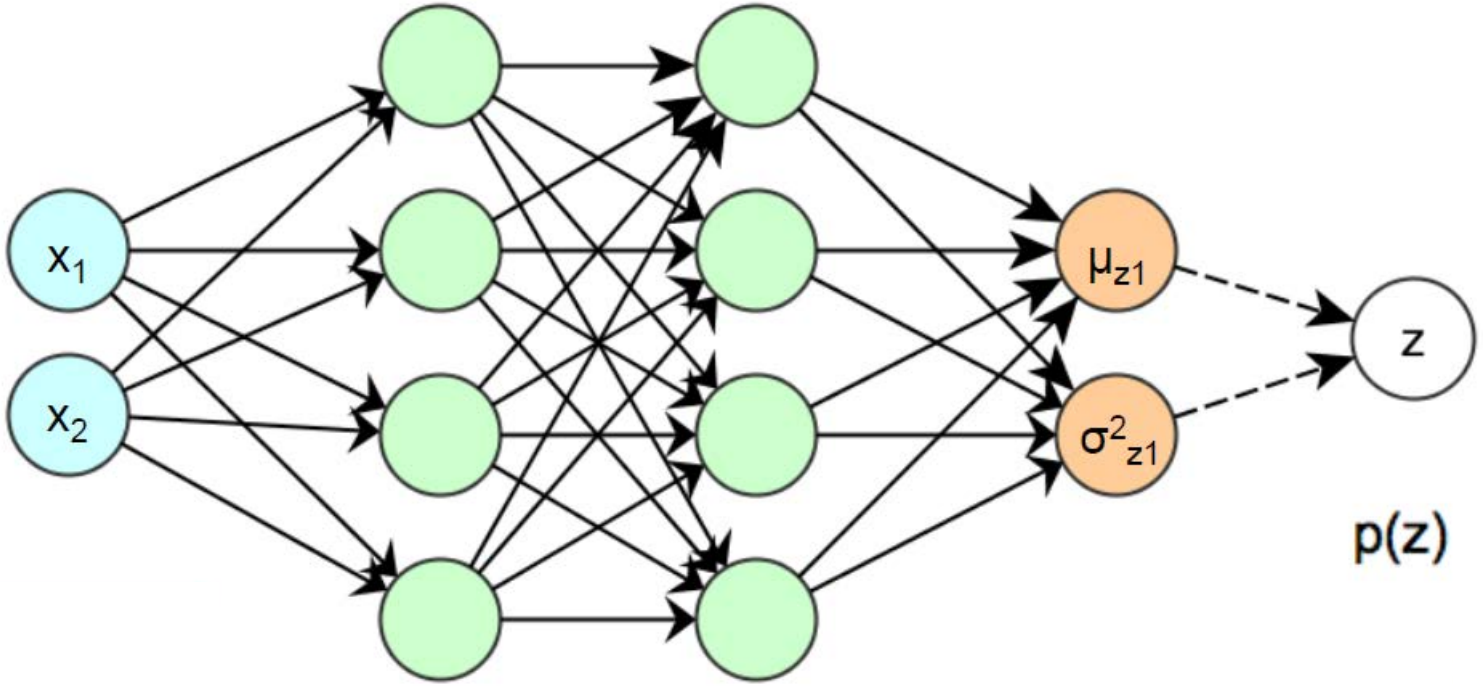
$$p_\theta(\mathbf{x}|\mathbf{z})$$



$$x|z \sim N(\mu_x, \sigma_x^2)$$

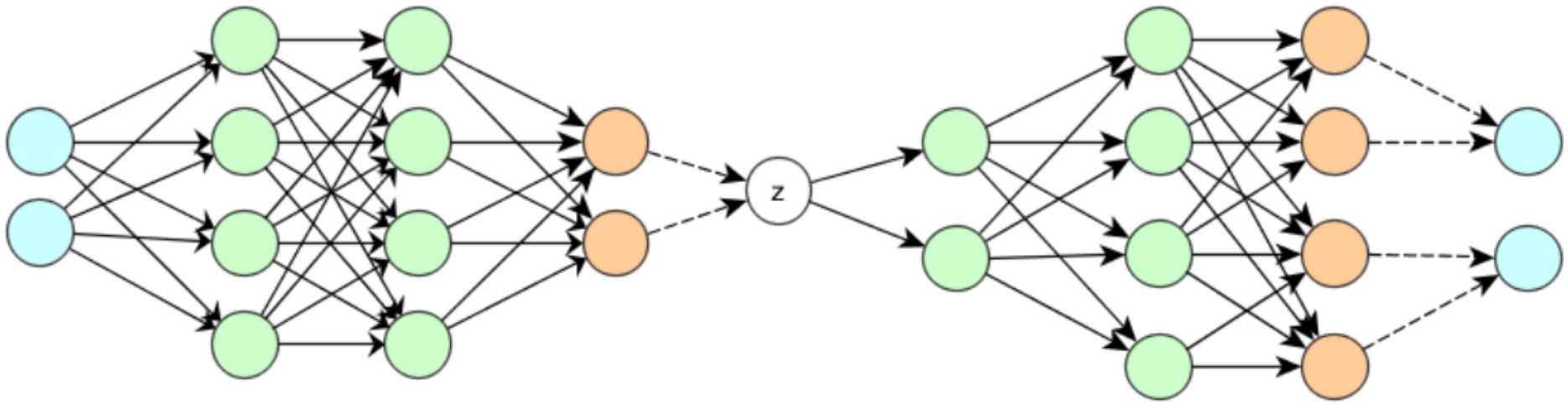
Model for Encoder

$$q_\phi(z|x)$$



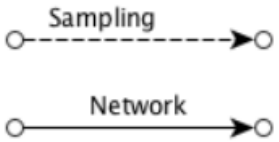
Putting it together: Autoencoder

$$q_\phi(z|x) \qquad\qquad p_\theta(x|z)$$



Learning the parameters φ and θ via backpropagation

Determining the loss function



Kullback – Leibler Divergence

- Measure between two distributions

$$D_{KL}(q(x)||p(x)) = \sum_{x \in X} q(x) \log \frac{q(x)}{p(x)}$$

$$\mathcal{D}\left[Q(z) \| P(z|X)\right] = E_{z \sim Q}\left[\log Q(z) - \log P(z|X)\right]$$

$$p_\theta(z|x) = P(z|X)$$
$$q_\phi(z) = Q(z)$$

Loss Function

- KL Divergence

$$\mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log Q(z) - \log P(z|X)\right]$$

- Applying Bayes Rule

$$\mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log Q(z) - \log P(X|z) - \log P(z)\right] + \log P(X)$$

$$\log P(X) - \mathcal{D}\left[Q(z)\|P(z|X)\right] = E_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}\left[Q(z)\|P(z)\right]$$

$$\boxed{\log P(X) - \mathcal{D}\left[Q(z|X)\|P(z|X)\right] = E_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}\left[Q(z|X)\|P(z)\right]}$$

Final Loss Function

Loss Function

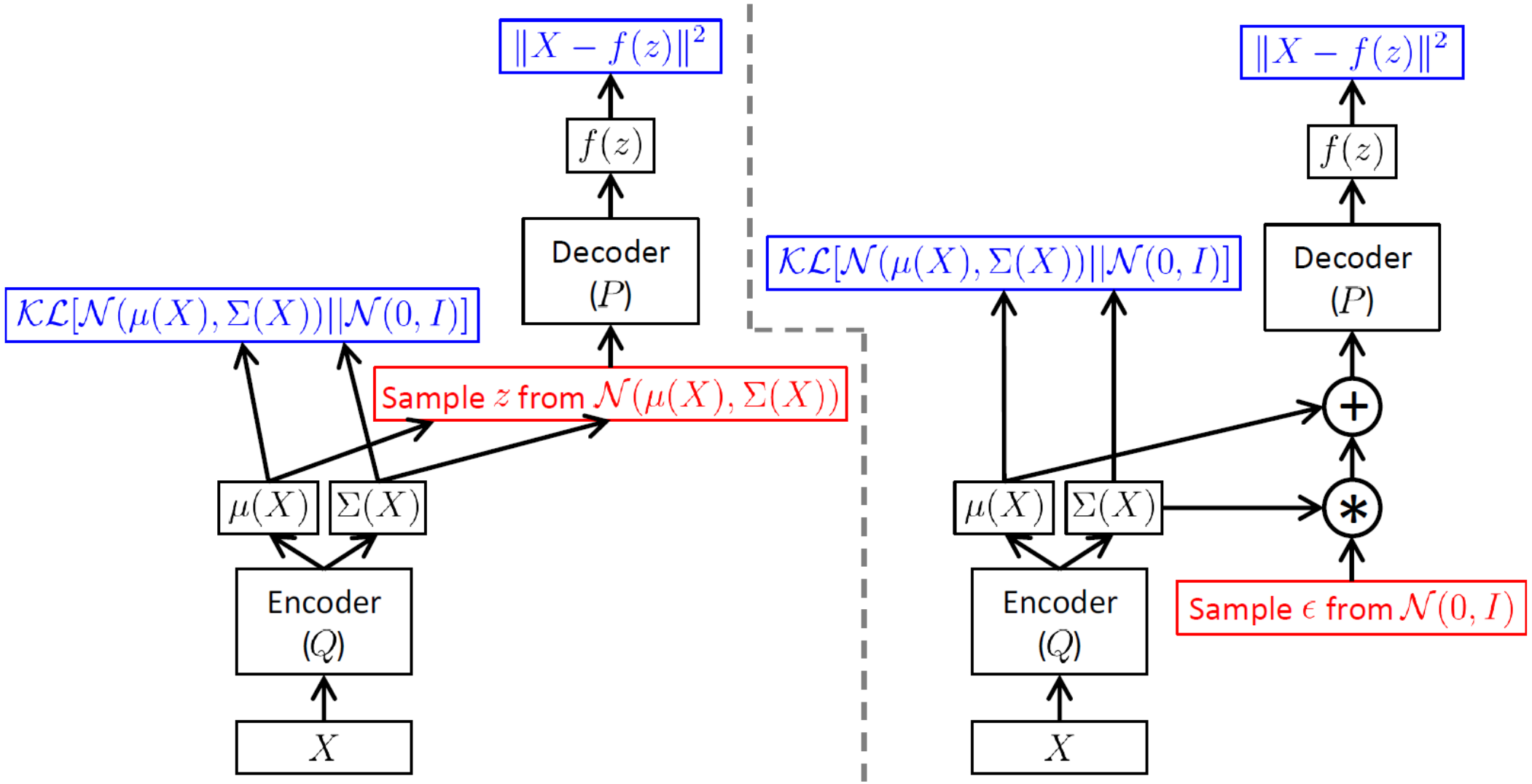
$$\underbrace{\log P(X) - \mathcal{D}\left[Q(z|X) \| P(z|X)\right]}_{\text{Quantity that we want to maximize}} = \underbrace{E_{z\sim Q}\left[\log P(X|z)\right] - \mathcal{D}\left[Q(z|X) \| P(z)\right]}_{\text{Can be optimized using SGD under some conditions}}$$

$$Q(z|X) = \mathcal{N}(z|\mu(X;\vartheta), \Sigma(X;\vartheta))$$

$$\mathcal{D}[\mathcal{N}(\mu(X), \Sigma(X)) \| \mathcal{N}(0, I)] = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma(X)\right) + \left(\mu(X)\right)^{\top}\left(\mu(X)\right) - k - \log\det\left(\Sigma(X)\right)\right)$$

Nice!
Depends only on X

Reparameterization Trick



Reparameterization Trick

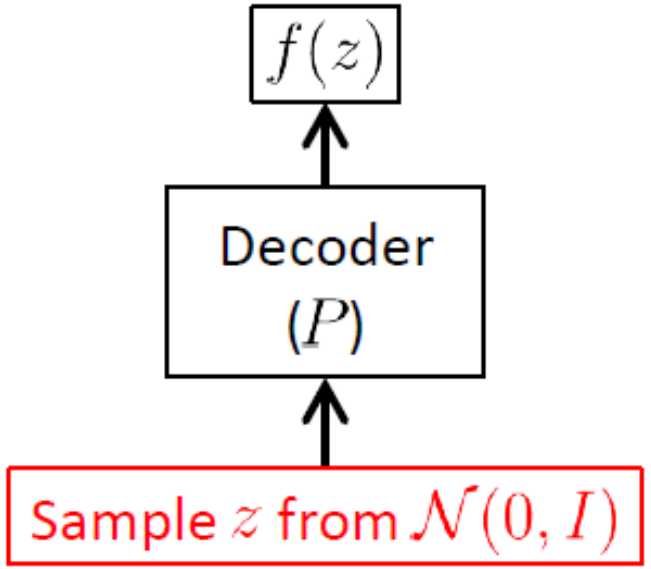
- Can be performed for a broad class of distributions, e.g.:

- **Location-scale transforms**
  - Normal, Laplace, Student t's, Logistic, etc.

- **Inverse of CDF**
  - Cauchy, Rayleigh, Pareto, etc

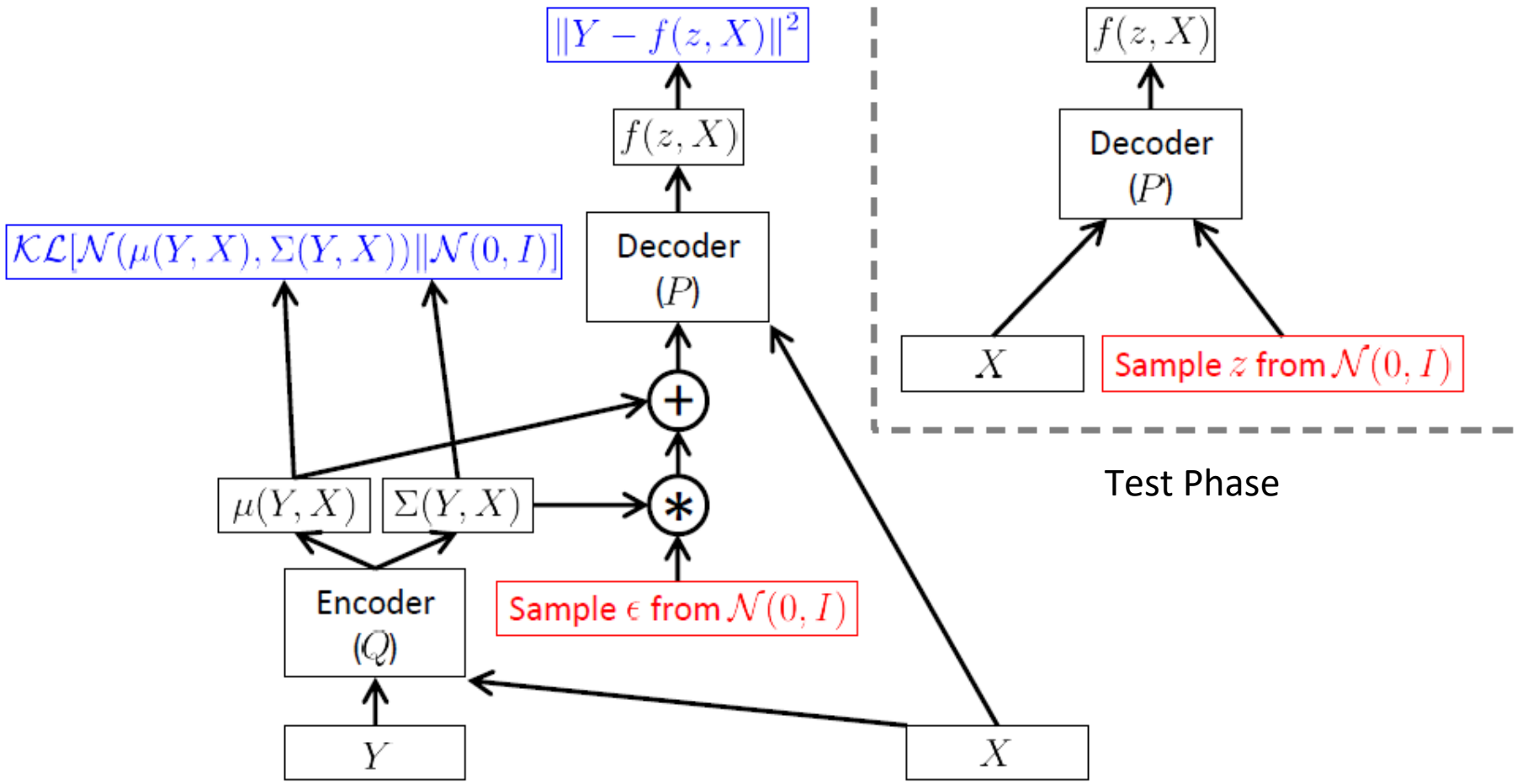
- Essentially: $Q(z|X)$ should be continuous

Testing



Applications of VAE variants

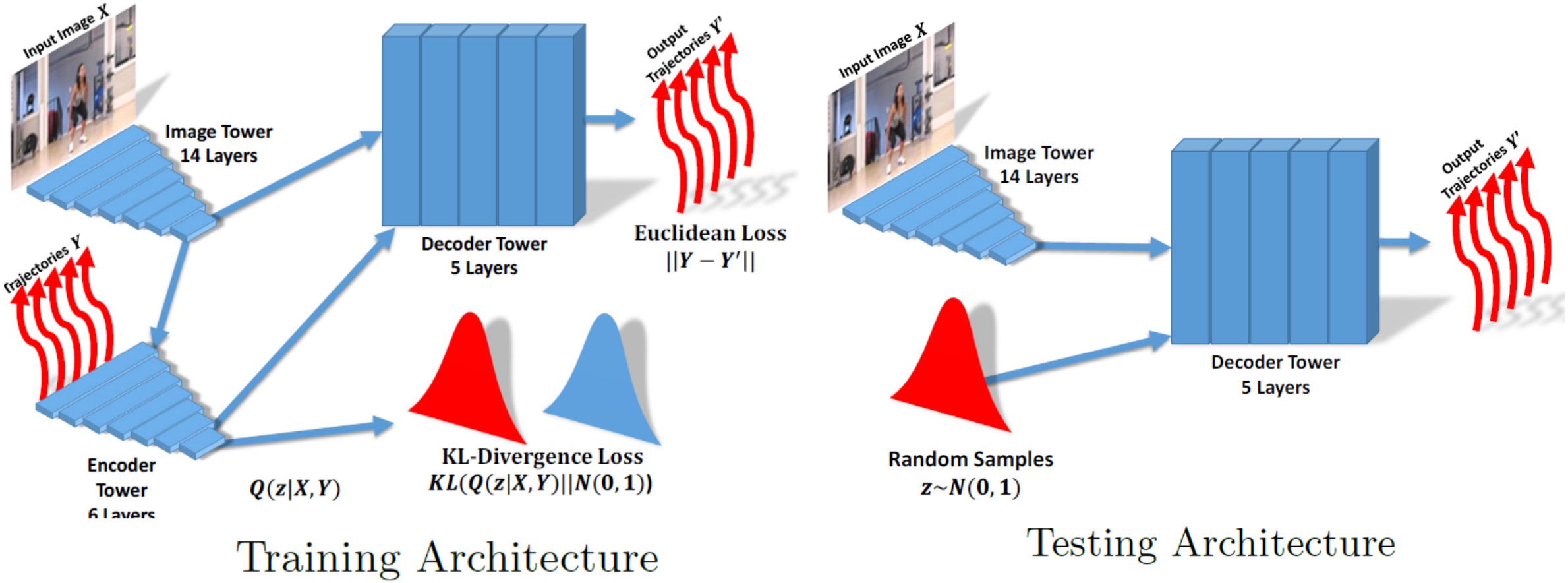
Conditional VAE



Test Phase

Training Phase

Forecasting from Static Images using CVAE



Walker et al., “An Uncertain Future: Forecasting from Static Images using Variational Autoencoders”, ECCV 2016

CVAE Results

Walker et al., “An Uncertain Future: Forecasting from Static Images using Variational Autoencoders”, ECCV 2016

CVAE Results

Figure 3: Visualization of generated samples with (left) 1 quadrant and (right) 2 quadrants for an input. We show in each row the input and the ground truth output overlaid with gray color (first), samples generated by the baseline NNs (second), and samples drawn from the CVAEs (rest).

Sohn et al., “Learning Structured Output Representation using Deep Conditional Generative Models”, NIPS 2015

CVAE Results

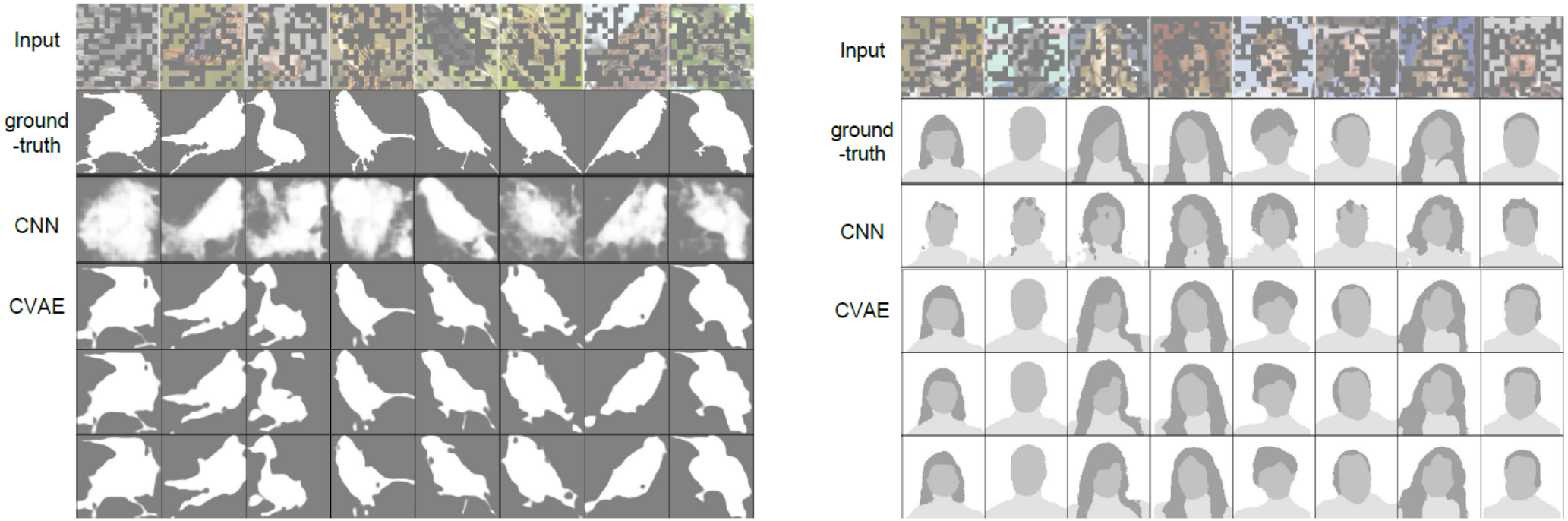
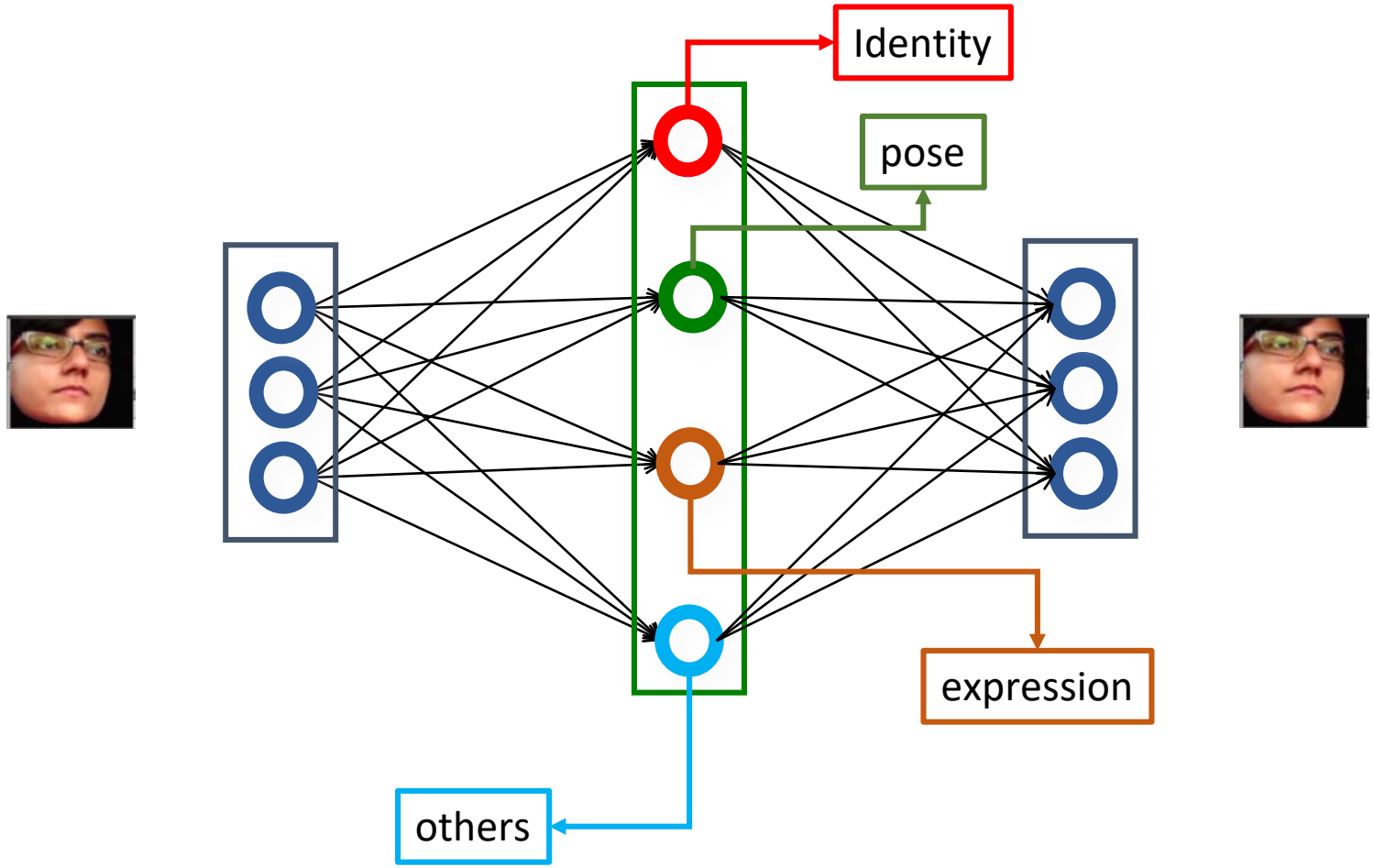


Figure 4: Visualization of the conditionally generated samples: (first row) input image with omission noise (noise level: 50%, block size: 8), (second row) ground truth segmentation, (third) prediction by GDNN, and (fourth to sixth) the generated samples by CVAE on CUB (left) and LFW (right).

Sohn et al., “Learning Structured Output Representation using Deep Conditional Generative Models”, NIPS 2015

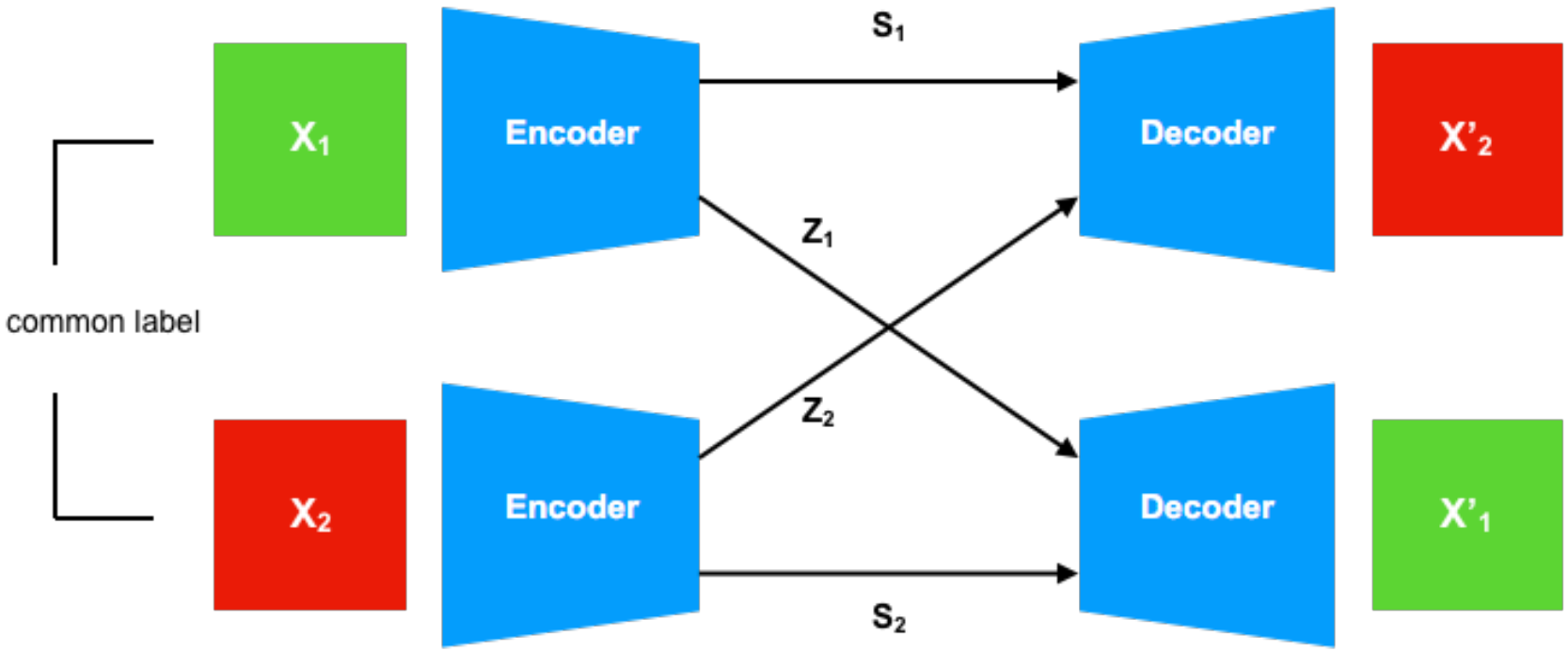
Disentangled Latent Representations using AEs



Disentangling Factors of Variation without Adversarial Training

Ananya

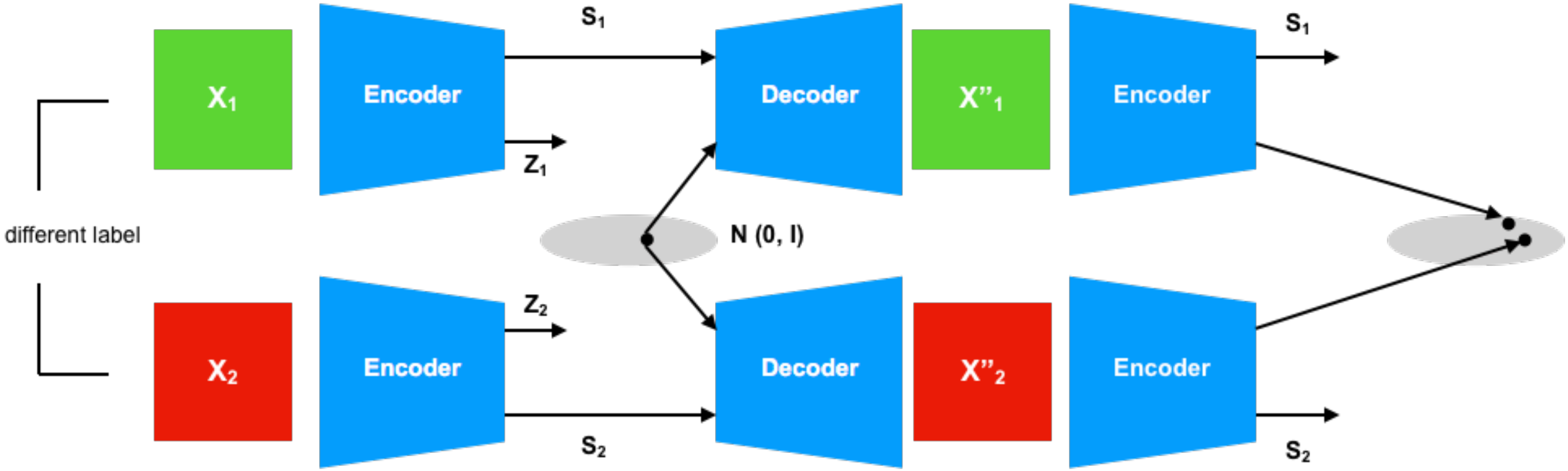
Cycle-Consistent VAEs – Forward cycle



S – specified latent space (e.g.,class labels)
Z – unspecified latent space (e.g., illum., pose, etc.)

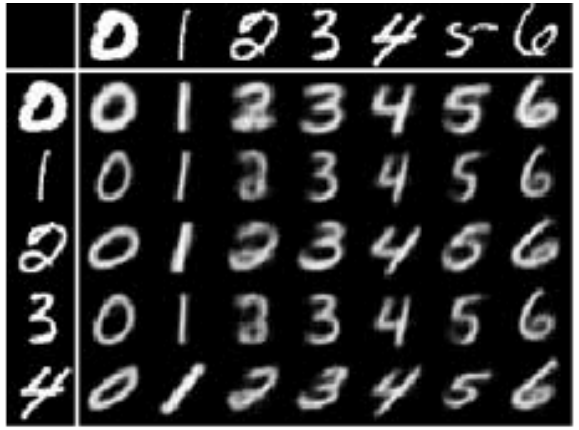
Jha et al, "Disentangling Factors of Variation with Cycle-Consistent Variational Auto-Encoders", submitted to ECCV `18

Cycle-Consistent VAEs – Reverse cycle



S – specified latent space (e.g.,class labels)
Z – unspecified latent space (e.g., illum., pose, etc.)

Initial results – comparison with previous non-adversarial method



(a) 16 dim unspecified factor space (b) 64 dim unspecified factor space

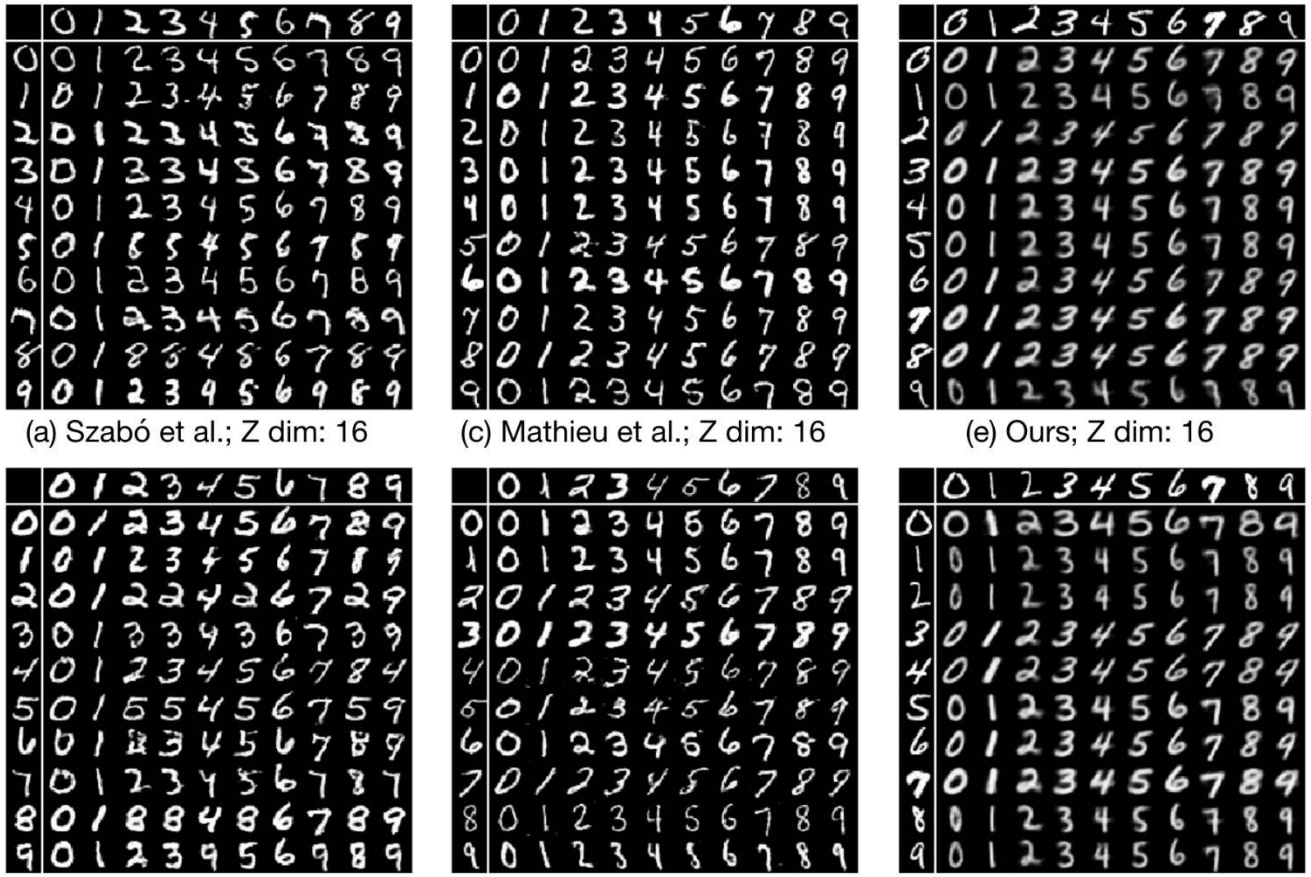
Interpolation in Latent Spaces

(a) MNIST

Generating Novel Views: Sampling the Style Space

(a) MNIST (b) Sprites (c) LineMod

Generating Novel Views: Swapping Style



(a) Szabó et al.; Z dim: 16

(c) Mathieu et al.; Z dim: 16

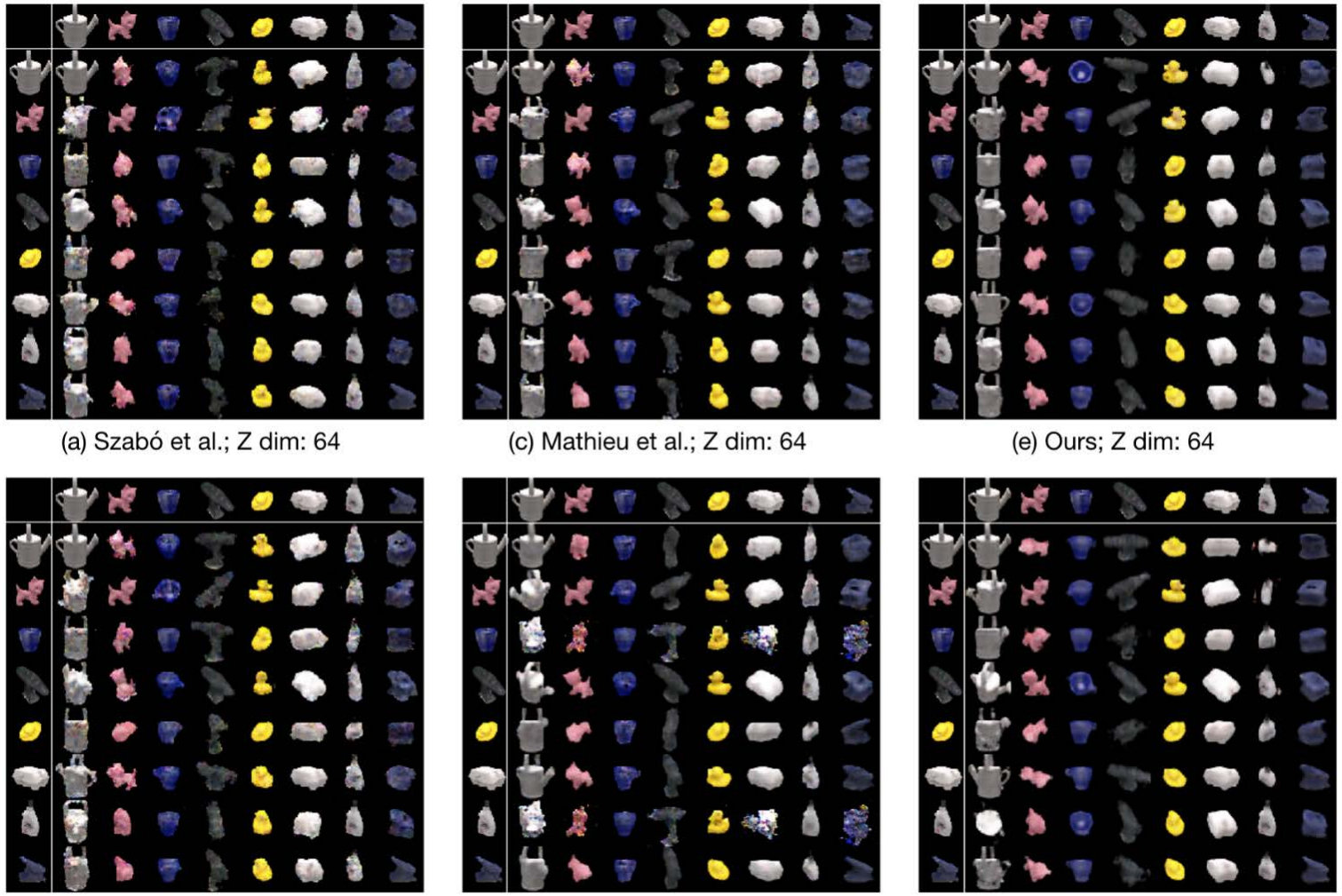
(e) Ours; Z dim: 16

(b) Szabó et al.; Z dim: 64

(d) Mathieu et al.; Z dim: 64

(f) Ours; Z dim: 64

Generating Novel Views: Swapping Style



(a) Szabó et al.; Z dim: 64

(c) Mathieu et al.; Z dim: 64

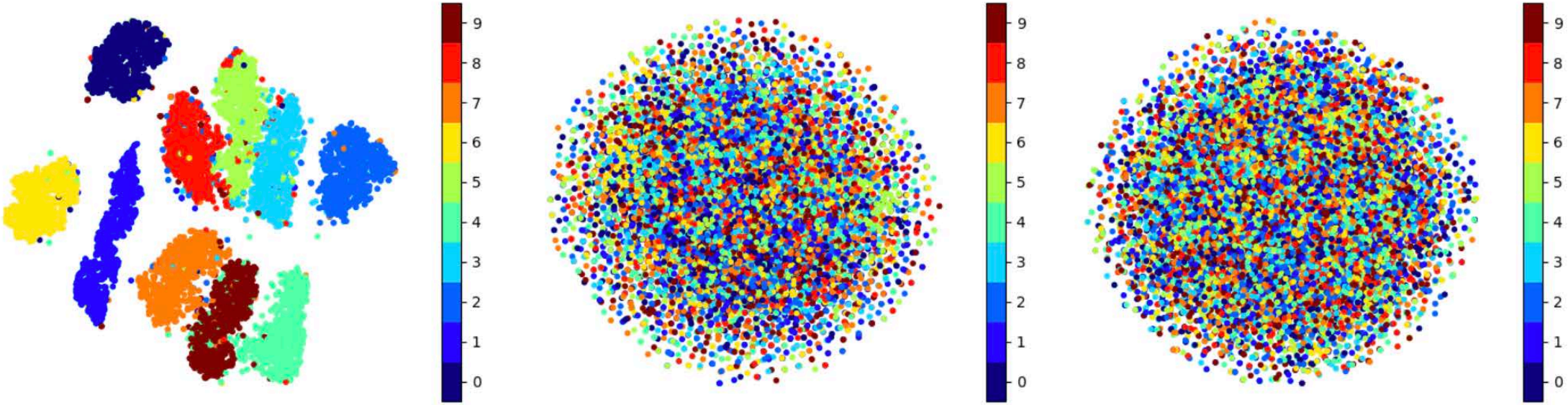
(e) Ours; Z dim: 64

(b) Szabó et al.; Z dim: 256

(d) Mathieu et al.; Z dim: 256

(f) Ours; Z dim: 256

Quality of Disentangling



(a) Szabó et al.; Z dim: 16

(b) Mathieu et al.; Z dim: 16

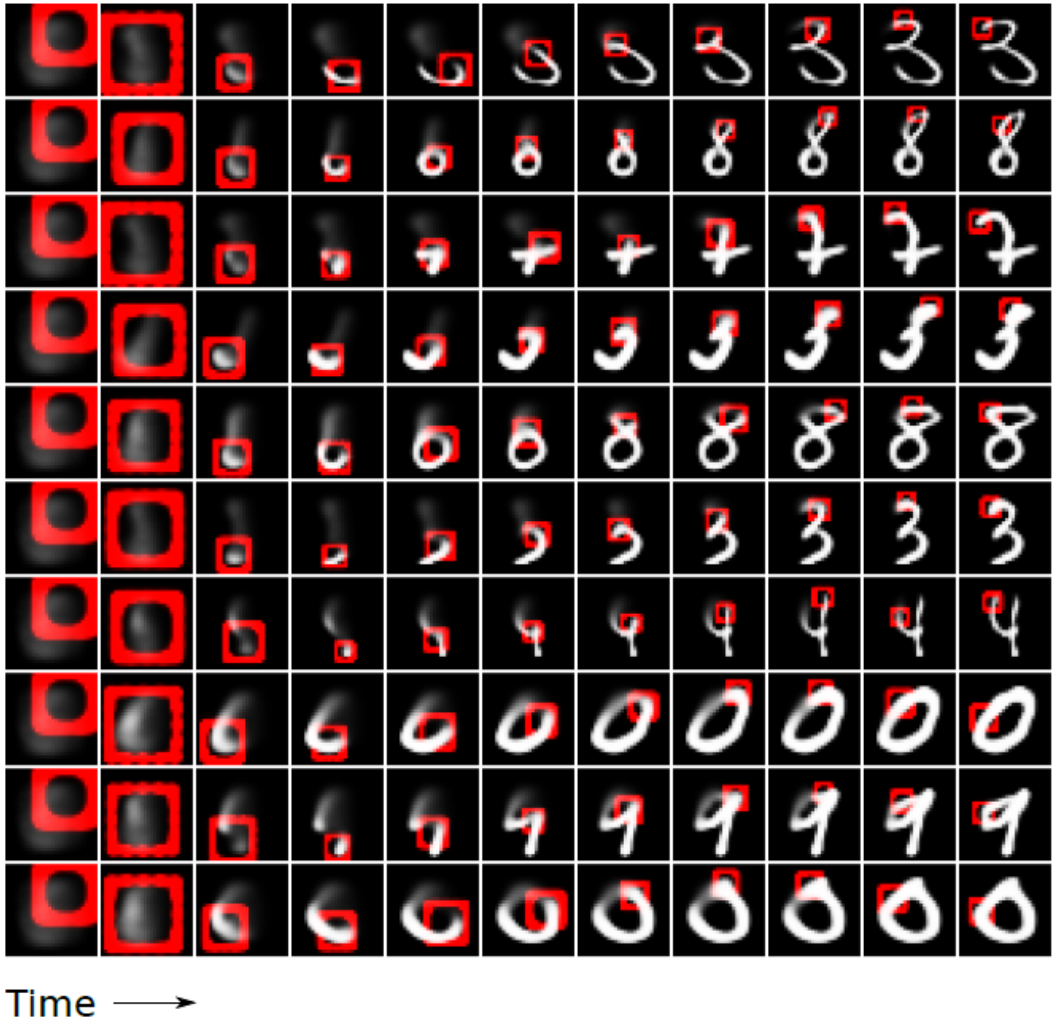
(c) Ours; Z dim: 16

Quality of Disentangling

Architecture	z dim	s dim	z train acc.	z test acc.	s train acc.	s test acc.
MNIST						
Szabó et al.	16	16	97.65	96.08	98.89	98.46
Mathieu et al.	16	16	70.85	66.83	99.37	98.52
Ours	16	16	**17.72**	**17.56**	99.72	98.35
Szabó et al.	64	64	99.69	98.14	99.41	98.05
Mathieu et al.	64	64	74.94	72.20	99.94	98.64
Ours	64	64	**26.04**	**26.55**	99.95	98.33
2D Sprites						
Szabó et al.	512	64	99.72	99.63	99.85	99.79
Mathieu et al.	512	64	12.05	11.98	99.18	96.75
Ours	512	64	11.55	11.47	98.53	97.16
Szabó et al.	1024	512	99.79	99.65	99.87	99.76
Mathieu et al.	1024	512	12.48	12.25	99.22	97.45
Ours	1024	512	11.27	11.61	98.13	97.22
LineMod						
Szabó et al.	64	256	100.0	100.0	100.0	100.0
Mathieu et al.	64	256	90.14	89.17	100.0	100.0
Ours	64	256	**62.11**	**57.17**	99.99	99.86
Szabó et al.	256	512	100.0	99.97	100.0	100.0
Mathieu et al.	256	512	86.87	86.46	100.0	100.0
Ours	256	512	**60.34**	**57.70**	100.0	100.0

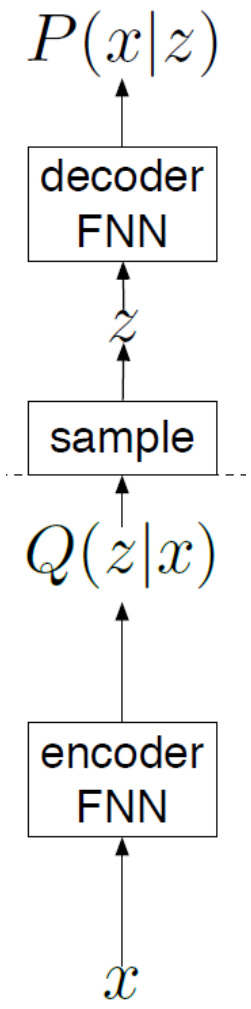
Deep Recurrent Attentive Writer (DRAW)

Combines the spatial attention mechanism with a sequential auto-encoding framework

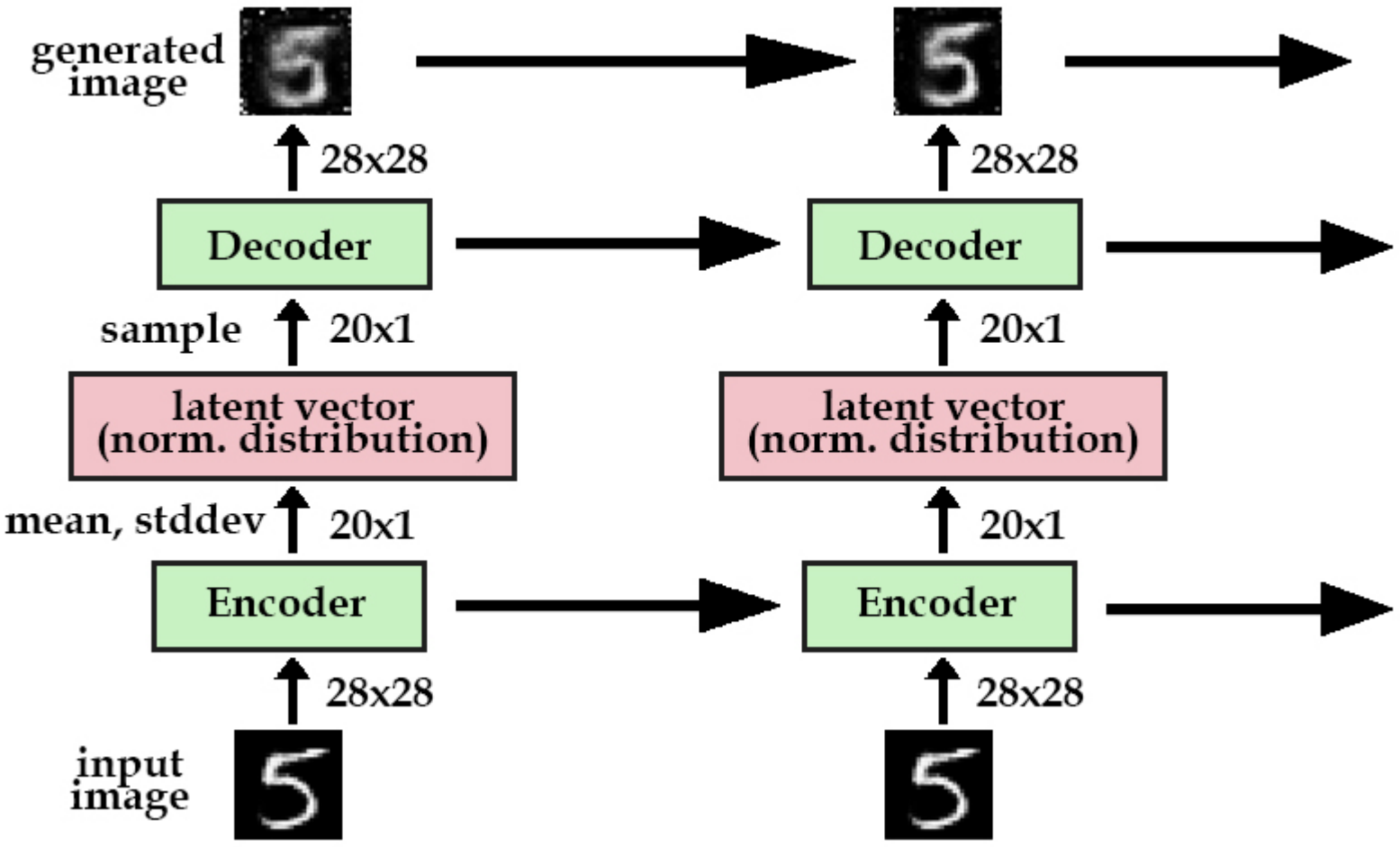


Gregor et al., “DRAW: A Recurrent Neural Network for Image Generation”, ICCV 2015 – by DeepMind

DRAW Architecture

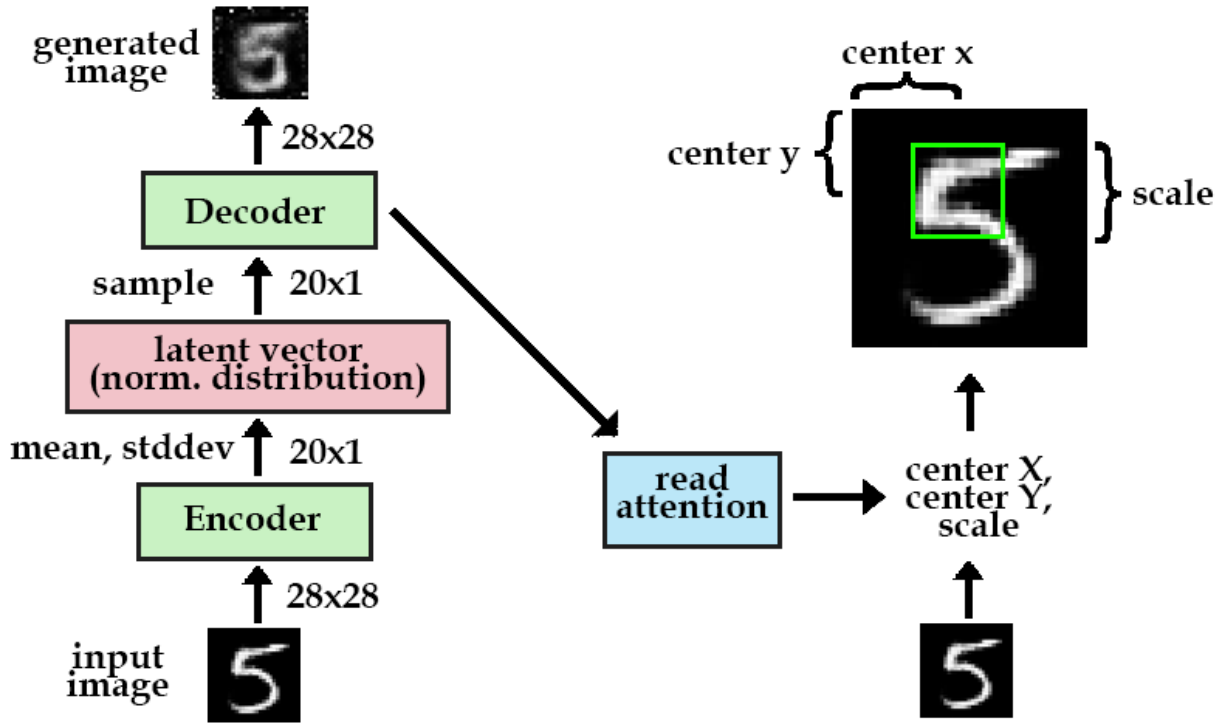


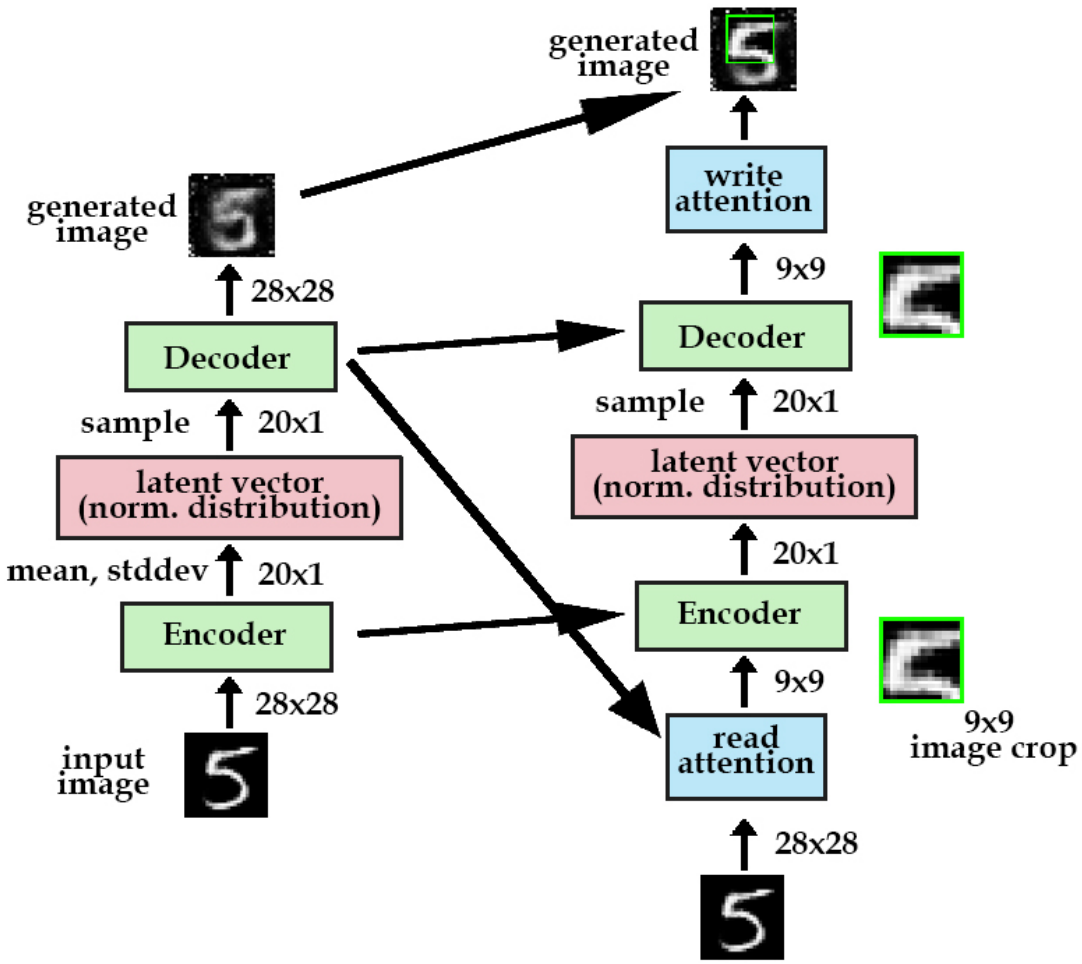
Conventional VAE



Draw Network

DRAW Architecture



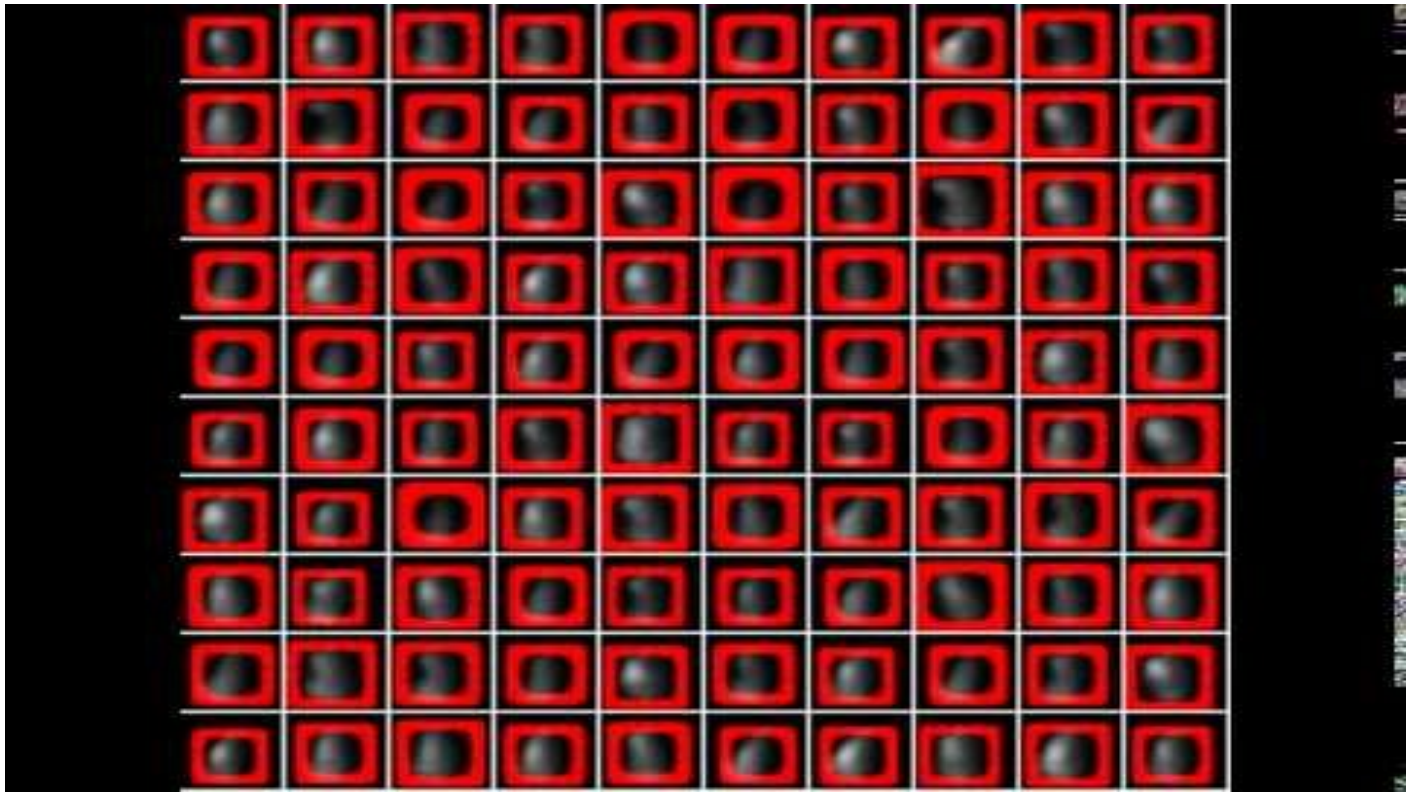


DRAW: Results on SVHN[1]

Time ⟶

[1] Street View House Numbers

DRAW Results on MNIST



VAE: Summary

- Variational Auto-Encoding: **scalable generative modeling**
  - Works with almost any model $p(x,z)$
  - Works with almost any approx. posterior $q(z|x)$
  - Scales to huge datasets
  - Just requires gradient ascent on single objective

VAE

- How to choose latent variables?
  - Don’t care – assume simple distribution and warp it through a sufficiently complicated function
- How to estimate the posterior distribution and sample from it?